

• Adlandırma kuralları

- önceden kullanılan ad tekrar kullanılmaz
- değişken adları 63 karaktere kadar içerebilir

ör

>> code = 443

>> mass-of-earth = 5.972E24

>> university = 'erciyes'

// Sonuna ; koyarak enterlayınca gösterime yapmıyor.

- ilk karakter harf olmalı
- ilk karakterden sonraki karakterler harf, sayı ve underscore(-) içerebilir.

ör

NetVal ✓ Left2Pay ✓ X-3 ✓ ERU1978 ✓

Net-Val X 1978ERUX %X X @Sign X

- MATLAB 'de büyük - küçük harf duyarlıdır. Num ile num farklıdır
- MATLAB'in anahtar kelimelerini kullanmaktan kaçının

ans → Default variable for ans

eps → en küçük artımsal sayı

pi → π

Inf → infinity

NaN → not a number, 0/0 gibi

i ve j karmaşık sayıları ifade eder ($\sqrt{-1}$)

- MATLAB'in anahtar fonksiyonları da değişken adı olarak kullanılmaz

namelengthmax → fonksiyonu değişken adının skip edebileceği max uzunluğu verir.

x = namelengthmax yapınca
63 veriyor.

Scalars

```
>> code = 443
```

```
code =  
443
```

```
>> c = 2.1 * 21 - 2 * a
```

```
c =  
42.1000
```

```
>> h = 22/7;
```

Çıktı Formatı

MATLAB sayıları floating-point değerler olarak saklar. ve onu öyle gösterir.

"format" ile gösterim değiştirilebilir.

```
>> format long
```

```
>> a = 1.123000123123123123;
```

```
>> a
```

```
a =
```

```
1.12300012312312
```

→ 16 karakter (noktayı da içerir)

```
>> a = 100004545.99923423499111;
```

```
>> a
```

```
a =
```

```
1.000045459992342e+008
```

→ 10⁸

```
>> format short
```

```
>> a = 1.123000123123;
```

```
>> a
```

```
a =
```

```
1.1230
```

→ 6 karakter

```
>> a = 100004545.99923423499111;
```

```
>> a
```

```
a =
```

```
1.0000e+008
```

→ 10⁸

Bunlar

format long	Kullanılan formatı öğrenmek için
format short	>> get(0, 'Format') yapılır.
format short Eng	
format long Eng	
format hex	alabilir

// Sayıyı basamaklandırırken ondalık kısma geçerken nokta(.) koy.

Vektörler

- MATLAB'de 2 tip array vardır
 - sayıların matrisi (double veya karmaşık)
 - cell array of objects (more advanced data structures)
- Row vector (sıra vektörü); bracket ([]) around the values with or without spaces.

```
>> row = [1 2 3 4];
```

```
>> row = [1, 2, 3, 4];
```

- Column vector (sütun vektörü); semicolon ile ayrılır.

```
>> col = [1; 2; 3; 4];
```

```
>> col = [1 2 3]';
```

→ transpose

// farklı şekilde.

- Vektörün boyutu: length

```
>> length(col)
```

```
ans =
```

```
3
```

* Bos vektor oluşturmak için;

```
>> v = [];
```

yapılır.

// Bezi vector Constructorları

• linspace()

>> a = linspace(0, 10, 5) // 0'dan 10'a kadar 5 elemanı yap

a =

0 2.5000 5.0000 7.5000 10.0000

• Colon operatörü (:)

[start: stepsize: end]

[0:0.25:1]

sekilde de yazılabilir.

>> m = 3:8, r = 0:0.25:1, s = 1:-1

m =

3 4 5 6 7 8

r =

0 0.2500 0.5000 0.7500 1.0000

s =

1x0 empty double row vector

• logspace (logaritmik olarak değerler üretilir)

Matrices

Vektörler gibi oluşturulur. (Farklı satırlara geçmek için ; kullanılır.)

>> A = [1 2; 3 4];

• Vektörleri birleştirerek (Concatenation)

>> r1 = [2 4];

m =

>> r2 = [3 6];

>> m = [r1; r2];

2 4

3 6 olur.

- vektör ve matrix birleştirildiğinde, boyutu ve tipi tutarlı olmalıdır.

$r3 = [2 \ 3 \ 4]$

$M = [r3; m] \rightarrow$ hatı verilir.

- Matrisin boyutu

$\gg [r, c] = \text{size}(m)$ $r=3$ $c=2$ olur gibi.
 ↳ satır ↳ sütun

$\gg r = \text{size}(m, 1)$ // satır sayısı

$\gg c = \text{size}(m, 2)$ // sütun sayısı

$\gg nd = \text{ndims}(m) \rightarrow$ Boyutlu dur.

//Özel Matrisler

- $\text{zeros}(m, n) \rightarrow m \times n$ boyutlu 0'lerden oluşan matris

- $\text{zeros}(n) \rightarrow n \times n$ boyutlu " " " "

- $\text{ones}(m, n) \rightarrow m \times n$ boyutlu 1 " " " "

- $\text{eye}(n) \rightarrow n \times n$ birim(identity) matris

- $\text{magic}(n) \rightarrow n \times n$ matris ama 1'den n^2 'ye kadar sayılardan oluşuyor ve satırların toplamı, sütunların toplamı eşit.

$\gg \text{eye}(3)$

ans =

1 0 0

0 1 0

0 0 1

$\gg \text{magic}(3)$

ans =

8 1 6

3 5 7

4 9 2

15 15 15

45=45

$\text{magic}(2)$

ans =

1 3

4 2

5 5

6+4=10

5+5=10

uniformly=düzenli olarak

0 0 0 0
0 0 0 0
6 0 0 0

// Rastgele matrisler

• rand(m,n)

• [0,1] aralığındaki rastgele sayıların düzenli dağıtıldığı m x n boyutlu matris oluşturur.

>> M = rand(2,3)

M = 0.8147 0.1270 0.6324
0.9058 0.9134 0.0975

• randn(m,n)

• rastgele sayıların normal olarak dağıtıldığı m x n matris. (mean=0, standart deviation 1)

>> M = randn(2,3)

M = -0.4336 3.5784 -1.3499
0.3426 2.7694 3.0349

→ her yazıldığında farklı oluşturur rand() de öyle.

• randi([min,max],m,n)

- [min,max] değerleri aralığından m x n'lik ayırık dağılmış matris oluşturur.

>> randi([5,10],2,3)

ans =
9 5 7
10 8 5

// Matrisleri Birleştirme ve Kopyalama

• repmat

>> b = repmat(5,3,2)

b =
5 5
5 5
5 5

>> b = repmat([1 2; 3 4],2,3)

b =
1 2 1 2 1 2
3 4 3 4 3 4
1 2 1 2 1 2
3 4 3 4 3 4

• dikey birleştirme (vertcat'e bak)

```
>> v1 = [2 3 4]; v2 = [1 2 3];
```

```
>> x = [v1; v2];
```

```
x =  
    2    3    4  
    1    2    3
```

• yatay birleştirme (horzcat'e bak)

```
>> v1 = [2; 3; 4]; v2 = [1; 2; 3];
```

```
>> x = [v1 v2];
```

```
x =  
    2    1  
    3    2  
    4    3
```

• vertcat ile

```
>> v1 = [2, 3, 4]; v2 = [1 2 3];
```

```
>> x = vertcat(v1, v2)
```

```
x =  
    2    3    4  
    1    2    3
```

• horzcat ile

```
>> v1 = [2; 3; 4]; v2 = [1; 2; 3];
```

```
>> x = horzcat(v1, v2)
```

```
x =  
    2    1  
    3    2  
    4    3
```

// Reshaping matrices

! operatörü

```
>> A = randi([1, 10], 4, 2)
```

```
A =  
     5     7  
    10     1  
     8     9  
    10    10
```


$\gg y = A(:)$ $\rightarrow A$ 'yi tek bir
 satırda yazıyor.
 $y =$
 5
 10
 8
 10
 7
 1
 9
 10

reshape

$\gg B = \text{reshape}(y, 2, 4)$
 $\% y$ 'yi 2×4 matrise dönüştürür.
 $B =$
 5 8 7 9
 10 10 1 10

$\gg M = \text{reshape}(\text{linspace}(11, 18, 8), [2, 2, 2])$
 $\% 11$ 'den 18 'e kadar 8 eleman oluştur ve bunu $2 \times 2 \times 2$ 'lik matrise koy.

$M(:, :, 1) =$ "1. boyutu"

11 13
 12 14

$M(:, :, 2) =$ 15 17
 16 18

Arrayleri çevirmek (flip) için birkaç fonksiyon bulunmaktadır.

- fliplr fonksiyonu matrisi soldan sağa çevirir.
- flipud yukarıdan aşağıya
- flip \rightarrow çevirir.

$\gg A = [-3, 0, 3; -2, 0, 2; -1, 0, 1]$

A =

-3 0 3
 -2 0 2
 -1 0 1

$\gg \text{flip}(A, 2) \% \text{ Soldan sağa (fliplr)}$

3 0 -3
 2 0 -2
 1 0 -1

1 5 9 13
 2 6 10 14
 3 7 11 15
 4 8 12 16

$\gg \text{flip}(A, 1) \% \text{ Yukarıdan aşağıya (flipud)}$

-1 0 1
 -2 0 2
 -3 0 3

olur.

13 9 5 1
 14 10 6 2
 15 11 7 3
 16 12 8 4

Vector Indexing

Matlab'ta indisler 0'dan değil, 1'den başlar.
 $v(n)$, v 'nin n . elemanını döndürür.

```
>> v = [-2 -1 0 1 2];
```

```
>> v(1)
```

ans =

-2

```
>> v(2:4) // 2'den 4. elemana kadar
```

ans =

-1 0 1

```
>> v(3:end) // 3'den sonuncuya kadar
```

ans =

0 1 2

- Index argümanı vektör olabilir. Bu durumda, her eleman bireysel olarak incelenir.

```
>> a = [-2, -1, 0, 1, 2];
```

```
>> x = [1 3 5];
```

```
>> a(x)
```

ans =

-2 0 2

```
>> a(1:2:5)
```

olabilir ya da // 1'den 5'e kadar 2

ser 2 ser

1, 3, 5 olur

-2 0 2 çıktısını verir

- Subscripts (sıra ve sütun)

```
>> M = [1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12; 13, 14, 15, 16]
```

```
>> M(1, 3) // 1. sıra 3. eleman
```

ans =

3

```
>> M(2, :)
```

ans =

5 6 7 8

```
>> M(2, 1:3)
```

ans =

5 6 7

```
>> M(2:end, 3)
```

ans =

7

11

15 olur.

• lineer indisler

>> M = [16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1];

>> m(2)

ans =

5

>> m(7)

ans =

7

1	16	2	3	13
2	5	11	10	8
3	9	7	6	12
4	4	14	15	1

• Subscripts ve lineer indisler arasında convertler.

>> ind = sub2ind(size(m), x, y) % Subscript → lineer indis

>> [x, y] = ind2sub(size(m), ind) % lineer ind → subscript

• Belli bir değerdeki yazdığımızdaki indisi bulma

>> L = [0, -1, 0; -1, 4, -1; 0, -1, 0];

>> ind = find(L < 0) // negatif elementlerin indisini denderiz

ind =

2

4

6

8

> = < = kullanılabilir.

>> ind = find(L > 0 & L < 5)

ind =

5

İşlemler

• Aritmetik işlemler (+, -, *, /)

>> 7/45

>> (2+i) * 4/5 => ans = 1.6000 + 0.8000i

• Exponentiation (^) (üstel)

```
>> (3+2*j)^2
```

ans =

24.000 + 10.0000i

• Karmaşıklık önlemek için parantez kullan

```
>> ((2+3)*3)^0.5
```

• Multiplication is not implicit + given parenthesis

```
>> 3(1+0.7)
```

↑
Hata verir.

Transpoz

Transpoz işlemi sütunları satıra, satırları sütuna çevirir.

ör

```
>> a = [-2; 2]
```

a =

-2 -1 0 1 2

```
>> a'
```

ans =

-2

-1

0

1

2

```
>> a = [3, 4; 1, 2; 3, 1]
```

a =

3 4

1 2

3 1

```
>> a'
```

ans =

3 1 3

4 2 1

Vektör İşlemleri

Matlabın içinde vektörlerde işlem yapmamızı sağlayan fonksiyonlar vardır.

```
>> a = [1 4 6 3]
```

a =

1 4 6 3

```
>> sum(a)
```

ans =

14

```
>> mean(a)
```

ans =

3.5000

```
>> var(a) // varyans
```

ans =

4.3333

>>std(z) // Standart deviation

ans =

2.0817

>>max(z)

ans =

6

>>min(z)

ans = 1

X • Verilen bir matrisde, bu işlemler matrisin her sütununda yapılır ve sonuç olarak bir satır vektörü döndürülür.

>>A=[1 2 3;4 5 6];

>>mean(A) // Her sütun için yapılır

ans =

2.5000 3.5000 4.5000

$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$

>>mean(A,2) // Satır satır işlem yapılır.

ans =

2

5

$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$

>>max(A) // Her sütun için

ans =

4 5 6

>>max(max(A))

ans =

6

X • zhs ve sign işlemleri matris veya vektörler üzerinde uygulanabilir.

signum'da $y = \text{sgn}(f(x)) = \begin{cases} 1 & f(x) > 0 \text{ ise} \\ 0 & f(x) = 0 \text{ ise} \\ -1 & f(x) < 0 \text{ ise} \end{cases}$

>> M = [0 4 -3; -1 0 2]

M =

0 4 -3

-1 0 2

>> abs(M)

ans =

0 4 3

1 0 2

>> sign(M)

ans =

0 1 -1

-1 0 1

Element-wise Operations (Element bazlı)

- Bu işlemler "element by element" yapılır.
- Bunları yapmak için, nokta kullanılır: \cdot $*$ $/$ $^{\wedge}$
- Eğer 2 vektör/matris toplanacaksa, çıkarılacaksa ya da element-wise çarpım veya bölüm yapılacaksa, aynı boyutta olmalıdır.

>> a = [1, 2, 3, 4]'; → transpose

>> b = [5, 6, 7, 8]'; →

$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \begin{bmatrix} 5 \\ 6 \\ 7 \\ 8 \end{bmatrix}$

>> a + b

ans =

6

8

10

12

>> a - b

ans = -4

-6

-4

-4

>> a.^2 //karesini alır

ans =

1

4

9

16

>> b.^2

ans =

25

36

49

64

>> 2.*b

ans =

5

12

21

32

>> 2./b

ans = 0.2000

0.3333

0.4286

0.5000

// Built-in fonksiyonlar

>> sqrt(2)

ans = 1.4142

>> log([1 2 3 4]) // element-wise logarithm

ans =

0 0.6931 1.0986 1.3863

>> exp([1 2 3 4]) // exponential

ans = 2.7183 7.3891 20.0855 54.5982

>> round([1.5 2; 2.2 3.1])

ans = 2 2

2 3

>> floor([1.5 2; 2.2 3.1])

ans = 1 2

2 3

>> ceil([1.5 2; 2.2 3.1])

ans = 2 2

3 4

Mantıksal ve Hissisel Operatörler

- Boolean değerler: 0 → false
nonzero → true

- Bazı lojik operatörler:

<, <=, >, >=

küçük, büyük, küçük eşit --

== ~=

eşit, eşit değil

&

mantıksal AND

|

mantıksal OR

~

mantıksal NOT

all

all true

any

any true

xor

xor

Mantıksal İndisleme

```
>> R = rand(5)
```

R =

• • • • •
•
•
•
•

} 5x5 random matrix oluşturulur.

```
>> R(R < 0.15)' // R'nin 0.15'ten küçük elemanlar alt indislerdeki  
ans =  
0.1270 0.0975 0.1415 0.0357
```

```
>> isequal(R(R < 0.15), R(find(R < 0.15)))  
ans =
```

1

Find Metodu

- find nonzero değerlerin indisini dönderir. Yeni bulduysa

- $\text{Index} = \text{find}(\text{condition})$

```
>> x = rand(1,10)
```

x =

0.4505 0.0838 0.2290 0.9133 0.1524 →
→ 0.8258 0.5383 0.8961 0.0782 0.4427

```
>> inds = find(x > 0.4 & x < 0.7)
```

inds =

1 7 10

```
>> x(inds)
```

ans =

0.4505 0.5383 0.4427 olur.

BÖLÜM 2: Introduction to MATLAB: Scripts and Functions, Control Loops and Data Structures

M-Files

MATLAB programı içeren text dosyaları şu şekilde açılabilir;

- command window
- m-files

There are 2 kind of m-files

- scripts
- functions

Scriptlerde fonksiyonlarda şimdiki komut dizileri tekler kullanılmama olarak şifler.

- Scripts are simple: komutları aynı command line'a yazdığınız gibi şifler.

- Fonksiyonlar çok esnek ve daha kolay genişletilebilir.

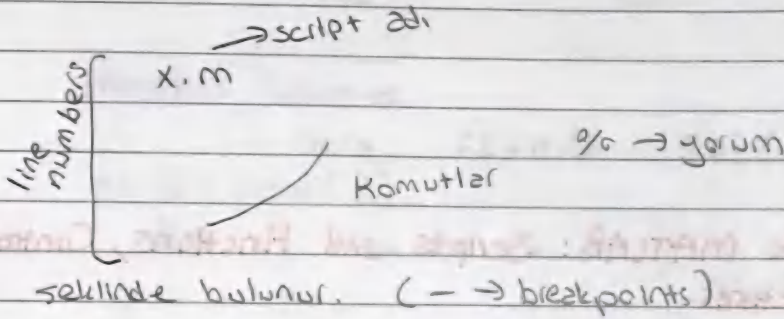
1) Scripts

Scriptler;

- bir dizi halinde oluşturulan komutlar koleksiyonudur.
- MATLAB editörde yazılmıştır.
- MATLAB dosyaları olarak kaydedilirler. (m uzantılı)

Command window'dan MATLAB dosyası oluşturmak için New Script'e tıklanır.

Scriptleri açmak içinse command window'dan
>> open helloworld.m



NOT

Yorum;

% ile yorum yapılabilir.

ilk gelen yorum scriptin help dosyasıdır.

uygun commentler zamanında tasarruf sağlar.

Scriptlerde bir input ve output yoksa, statik diyebiliriz.

Bir script dosyasında oluşturulan ve değiştirilen tüm variables, çözümleri durdurduktan sonra bile çalışma alanında bulunur.

→ workspace

Scriptler workspace deki bir degiskene ellip islemler yapabilir.

Ör

coin.m adında bir script yazın. Bu scriptle yazı-tura oyunu izleyebilirsiniz.

Her bir atışta (toss) Heads veya Tails yazmalı.

% coin.m

% a script that flips a coin and displays the output.

if rand < 0.5 % random sayı 1/2'den küçükse

display('Heads');

else

display('Tails');

end

NOT

• Burada scriptlerinizin başka kişiler veya kendiniz tarafından daha kolay anlaşılmasını sağlayacak bazı ipuçlarını göreceğiz.

• Script dosyasları header bölümünde sunulan içermelidir.

- program ne yapacak

- yazan kim

- kimin tarafından yazıldı veya gözden geçirildi

- Variable dictionary

• Programın anlaşılabilir olması için white space'ler kullan.

(Kodlar arasına enter'lar)

• Commentler kullan.

• Degiskenler için anlamlı ifadeler kullan. ^{ör} image-size

• clear all ve clc komutlarını scriptin başında kullan.

2) Functions

- Fonksiyonlar scriptlere benzer, tek farkı içinde fonksiyon tanımlaması olmasıdır.

function output = functionName(input)

- Fonksiyon içinde oluşturulan ancak döndürülmeyen (return edilmeyen) tüm değişkenler, işlevin gelişmesi durduktan sonra kaybolur.

ör

function [avg, sd, range] = stats(x)

help

← %stats: ortalama, standart sapmayı ve verilen range'i hesaplar.

avg = mean(x);

sd = std(x);

range = [min(x); max(x)];

end

MATLAB'de fonksiyon yazmak için:

dosya adı ile
aynı olmalı

cmd'ye save fonksiyon.m yap ya da new → function file yap

fonksiyon çağırma adı da aynı olmalı

bunu yaz ancak stats yerine fonksiyon yaz.

cmd'den x = [3, 2] yap

fonksiyon(x) ile çağırılabilir.

User - Defined fonksiyonlar

must have
the reserved word
function

function name
should match m-file
name.

Inputs must be
specified

function [out1, out2] = function Name (in1, in2)

birden fazla

output bracket içinde belirtilmeli

- Return 'e gerek yok. = MATLAB zaten fonksiyon tanımındaki yapılan değişken isimlerini dönecektir.
- Variable scope = Fonksiyon içinde oluşturulan ancak döndürülmeyen (return edilmeyen) tüm değişkenler, işlevin çalışması bittikten sonra kaybolur.

overloading

- MATLAB fonksiyonları genellikle overloaded tir.

- Değişken sayıda input alabilir.
- Değişken sayıda output dönebilir.

>> a = zeros(2,4,8); → 2x4 boyutunda 8 tane matris oluştur

>> D = size(a) → 2,4,8 döndürür. Elementleri 0 olsun

>> [m,n] = size(a) → m=2 n=32 döndürür.

>> [x,y,z] = size(a) → x=2 y=4 z=8 döndürür.

>> m2 = size(a,2) → 4 döndürür

2,1 → 2

2,3 → 8 döndürür.

- Kendi fonksiyonlarınızı bu şekilde değişken sayıda input ve output-
larla overload edebilirsiniz.

Number of Inputs / Outputs

- nargin, geliştirilen fonksiyondaki input değişkenlerinin sayısını döndürür.

```
function c = addme(a,b)
```

```
    switch nargin
```

```
        case 2
```

```
            c = a + b;
```

```
        case 1
```

```
            c = a + a;
```

```
        otherwise
```

```
            c = 0;
```

```
    end
```

```
end
```

Command Prompt

```
addme(3,4)
```

```
7
```

```
addme(3)
```

```
6
```

```
addme()
```

```
0
```

```
dur.
```

- nargout, geliştirilen fonksiyondaki output değişkenlerinin sayısını döndürür.

Global Variables

- Fonksiyonlar, temel workspace'i kullanmazlar. Her fonksiyon kendine alt workspace'i vardır.
- Birkaç fonksiyon tek bir değişkeni paylaşmışsa gerekliyse, bu değişken global olarak tanımlanabilir.
- Herhangi bir fonksiyonda bu global değişkenin değiştirilmesi, diğer fonksiyonlara da yansır.

global x
x = 5
CMD'de

function r = displayGlobalX

global x
val = x;

disp(val);

end

buraya gellesek de orada değeri

Anonymous Functions

- Dışarıya oluşturulmadan fonksiyonlar
 - 0 veya fazla parametre alabilir.
 - İçeride anonymous fonksiyonlar için verilir.
 - Expression ve gerekli ifadeler gerekli.

CMD

>> function = @(x,y) x^2 + y^2;

fonsiyon adı almalı,

Değer sonra

>> fonsiyon adı (3,4)

= 25 olur.

Local Functions

- MATLAB dosyasında birden fazla fonksiyon bulunabilir.
- İlk fonksiyon main fonksiyondur.
 - Herhangi bir yerden çağırılabilir.
- Other functions local fonksiyonlardır.
 - Only callable from main function or other local functions in same file.
 - Enables modularity (large number of small functions) without creating a large number of files.
 - Unfavorable (elverişsiz) from code reusability point.

Control Loops

→ relational

İlişkisel (Relational) ve Logical Operatörler

* Boolean değerlerde = zero = false
non zero = true

Bazı mantıksal operatörler

Bazı ilişkisel Operatörler

$<, <=, >, >=$ } less than, less than
equal to etc.
 $=, \sim$ } equal to, not equal to

$\&$ → logical and
 $|$ → logical or
 \sim → logical NOT
all → all true
any → any true
xor → xor

false
find
is logical
logical
true gibi

if / else / ~~if~~ else if

if expression1
statements1

elseif expression2
statements
--

else
statements--

end

- Paranteze gerek yok; comment blocks are between reserved words.

switch

Switch variable

case value1

statements--

case value2

statements--

otherwise % for all other values

statements

end

for

- iterasyon sayısı biliniyorsa kullanılır.

for n=1:100

statements--

end

loop değişkeni:

- vektör gibi tanımlanır.
- is a scalar within the command block.
- ardışık aralık değer vermek
gerekli değildir
(fakat diziye atılır olur.)
consecutive = ardışık

while

- while daha geneldir. İterasyon sayısını bilmeye gerek yok.

```
while condition
    commands (true iken)
end
```

Pre-Allocation of Memory

İteratif olarak array boyutunu arttıran for ve while döngüleri; performansı olumsuz etkileyebilir.

- Tekrarlı olarak array boyutunu değiştirmek MATLAB'de extra time harcamasını gerektirir.

```
for n = 1:10000
    x(n) = rand();
end
```

Not:
x = zeros(1, 10000);
for n = 1:10000
 x(n) = rand();
end

Preallocation yapmadan boyut değiştirirsek;

```
x = 8;
x(2) = 10;
x(3) = 12;
```

8	x(2)=10	8	x(3)=12	0x0000
0x0008	→	8	→	8
0x0010		10		10
0x0012		0x0012		8
		" 14		10
		" 16		12
		" 18		0x0018

Gök fazla bellek alanı harcar. Gereksiz veri tutar.

Preallocation yaparsak;

→ $x = \text{zeros}(1,3);$ → 1×3 matris
 $x = 8$
 $x(2) = 10;$
 $x(3) = 12;$

8	8	8
0	10	10
0	0	12

olun

try-catch

try
Statements
catch exception
Statements
end

ör
try
 $z = \text{noteFunction}(5,6)$
catch
warning('Format using function.
Assigning a value of 0.1');
 $z = 0;$
end

Remarks (Açıklamalar)

- break: immediately jumps execution to the first statement after the loop.
- continue: loop'da geri kalan statementleri atlar ve next iterasyondan başlar.
- return: immediately end a functions routine.
- eğer çok complex değerler varsa bunlar için i ve j kullanma
- Loops are inefficient in MATLAB. Try using built-in functions instead
- Allocating memory before loops greatly speeds up computation times

ör Avoiding loops

$x = \sin(\text{linspace}(0, 10 * \pi, 100))$ verilsin. Kaç tane entry positif
⇒ 0'dan $10 * \pi$ 'ye kadar 100 eleman oluşturun.

Using a loop and if/else

count=0

for n=1:length(x)

if x(n)>0

count=count+1;

end

end

Daha akıllıcası şu olur.

count = length(find(x>0));

Yeni bazen looplardan vazgeçilmelidir. Built-in fonksiyonlar bunu daha hızlı yapabiliyor.

Timing

- tic/toc : kodun çalıştırıldığında ne kadar zaman aldığı gösteren built-in fonksiyondur.
- Tic timerı açar ve toc ile zaman hesaplanır ve sonuç yazdırılır.

tic

mySum = 0;

for n = 1:10000

mySum = mySum + n;

end

toc

- MATLAB'ın Profiler ile kodların çalışma zamanında detaylı rapor oluşturulur.

- Bunu ise `profile on`, `profile off` ve `profile viewer` metoduyla yapabiliriz.

```
>> profile on
```

```
>> test_for
```

```
Elapsed time is 0.039972 seconds
```

```
>> profile viewer
```

```
>> profile off
```

Data Structures

- 2D' matrisler kullandık.

- n boyutlu olabilir.

- her eleman aynı tipte olmalı. (integers, doubles, characters...)

- Matrisler space-efficient ve convenient for calculation.

- Bazen çok karmaşık veri tipleri duruma göre daha uygun olabilir.
(appropriate=uygun)

- Cell array = array'e benzer, ancak elemanları aynı veri tipinde olmak zorunda değil.

- Structs: variable adını ve değerlerini tek yapıda toplar.

1) Cell Arrays

Cell matrix gibidir, fakat her alan herhangi birşey olabilir.
(hatta matris bile)

2x2 matrix

1.2	2.3
3.2	2.1

3x3 Cell Array

John

Mary

Leo

	32	
	27	1
	18	

2
4

[]

Bu şekildeki gibi bir cell array, yazını ve sayıların yazını tutabilir.

To do same with matrices, you would need 3 variables and padding

Veri tipini initializationda

- To initialize a cell, specify the size
3 rows 10 columns
>> a = cell(3,10);

- ya da manuel olarak yazmak için, curly braces {} kullanılır.
→ 1 row 3 column için
>> c = {'hello world', [1 5 6 2], rand(3,2)};

- bir celldeki eleman her şey olabilir.
elementlere erişmek için;
>> a{1,1} = [1 3 4 -10];
>> a{2,1} = 'hello world';
>> a{1,2} = c{3};

2) Structs

- Structlar isim eklemenizi sağlayan ve ilgili değişkenleri saklar
 - C' deki gibi: objeler var.
- To initialize an empty struct
 - >> s = struct([]);
 - size(s) 1x1 olur.
- Initialization is optional but is recommended when using large structs
- To add fields
 - Field değerleri herhangi bir şey olabilir: matrix, cell hattı struct
 - useful for keeping variables together.
 - >> s.name = 'Aslı Hızım';
 - >> s.scores = [80, 60, 70];
 - >> s.status = 'P';
 - ↳ substring kullan s(1).adi - gibi

cell2mat - cell2struct

- cell2mat => convert cell array to ordinary array of the underlying data type.

>> C = {[1], [2, 3, 4]; [5; 9], [6 7 8; 10 11 12]}

C =

2x2 cell array

{ [1] } { 1x3 double }
{ 2x1 double } { 2x3 double }

>> A = cell2mat(C)

olur.

A =

1 2 3 4
5 6 7 8
9 10 11 12

• cell2struct = convert cell array to structure array.

>> c = {'001', 'Aslı', 75; '002', 'Hızım', 85};

>> fields = {'number', 'name', 'mark'};

>> cStruct = cell2struct(c, fields, 2)

2 eleman

cStruct =

2x1 struct array with fields:

number
name
mark

ÖRNEKLER

Practice - 2 - 1. m

$A = \text{ones}(10, 10);$ \rightarrow 10×10 boyutunda birden oluşan matrisler.

$B = \text{ones}(10, 10);$

$v = 1:10;$ % 1'den 10'e kadar sayıları tutar.

$[m, n] = \text{size}(A);$

$\rightarrow_{10} \rightarrow_{10}$

for $i = 1:m$

$A(i, :) = A(i, :) - v;$

end

1. satırın

tüm elemanları

1. satırın

tüm elemanları

Yani satırcı çıkarma yapıyor.

sonuç

$A = \begin{bmatrix} 0 & -1 & -2 & -3 & -4 & -5 & -6 & -7 & -8 & -9 \\ 0 & -1 & -2 & -3 & -4 & -5 & -6 & -7 & -8 & -9 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & -1 & -2 & -3 & -4 & -5 & -6 & -7 & -8 & -9 \end{bmatrix}$

10 tane

$B = B - \text{repmat}(v, m, 1)$ % 10×1 boyutunda elemanları v 'den oluşan matris

Yani 10×10 boyutunda olmuştur.

B 'de aynı çıktıyı vermiş olacaktır.

Practice-2-2.m

A = randi([-10,10], 10, 10); 10x10 boyutunda elementler.
-10 ile 10 arasında matris.

[m,n] = size(A);

B = zeros(m,n);

C = zeros(m,n);

% % A'daki pozitif elementleri B'ye atalım.
section 2. yöntem

for i = 1:m

for j = 1:n

if A(i,j) > 0

B(i,j) = A(i,j);

end

end

end

Run section diyerek bulması
çalıştırılabilir.

% 2. yöntem

% % 2.

ind = find(A > 0);

C(ind) = A(ind);

D = zeros(m,n);

% % 3. yöntem

D(A > 0) = A(A > 0); % Copies into D only the elements of A
that are > 0

if isequal(B, C, D)

disp('B, C ve D eşitler')

end

Practice - 2-3.m

% Verilen bir dizinin elemanlarını (1xN vektör) aynı boyuttaki
% diğer vektöre ardışık elemanların toplamıyla eklenmesi
% İlk eleman aynı kalacak

$v = \text{randi}([1, 100], 1, 10);$ $\rightarrow 1 \times 10$
[m,n] = size(v); $\rightarrow 3 \ 2 \ 1 \ 4 \ 2$ olsun
w = zeros(m,n); $\rightarrow 1 \times 5$ bu
u = zeros(m,n);

% % for ile

for i = 1:n

if i == 1

w(i) = v(i);

else

w(i) = v(i-1) + v(i);

end

end

% % vektöre.

$w = [0 \ v(1:\text{end}-1)] + v;$ $\rightarrow w = 3 \ 5 \ 3 \ 5 \ 6$ olur.

elementer
söz konusu gibi olur.

if isequal(w,u)

disp('u and w are equal');

end

\rightarrow eşit olurlar.

BÖLÜM 3: GRAPHICS, VISUALIZATIONS AND SYMBOLIC TOOLBOX

Basic Plotting (Temel Grafiklendirme)

`plot()` her $(x; y)$ çifti için nokta oluşturur ve sonrasındakilerle line ile bağlar.

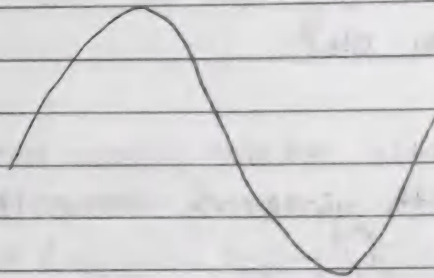
`>> x = linspace(0, 2*pi, 1000);`

→ 0'dan 2π 'ye kadar 1000 tane sayı oluşturulur

`>> y = sin(x);`

→ Bu fonksiyonu çizdirecektir

`>> plot(x, sin(x))`



şeklinde bir çıktı olur.

`>> plot(y);` 0'dan 1000'e kadar gösterir.

→ x ve y aynı uzunlukta olmalıdır.

• `figure`: to open a new figure and avoid overwriting plots.

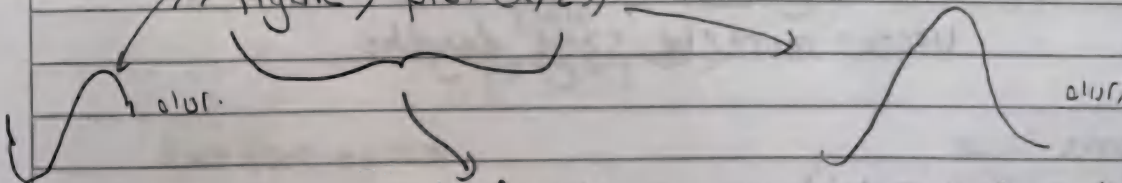
`>> x = [-pi : 0.1 : pi];` // -pi'den pi'ye kadar 0.1 ilerleyerek vektör oluştur.

`>> y = sin(x);`

`>> z = cos(x);`

`>> plot(x, y);` %o automatically creates a new figure

`>> figure; plot(x, z);`



Bununla x, y çizilmeye devam ederken x, z çizilmi de başka pencerede çıkar.

• close: close figures

>> close 1

>> close 211

→ 1 tane kepe (en

hepsini kepe

ilk

çilen)

• hold on/off: multiple plots on same graph

>> plot(x,y); hold on

>> plot(x,z,'r'); hold off.

Buton x bazarak da olabilir.

>> plot(x,y)

>> hold on

>> plot(x,z,'r')

>> hold off

>> plot(x,y) → Sadece bu olur.

red

ikisi aynı figure penceresinde gösterilir.

• Plot fonksiyonunu daha smooth göstermek isterseniz, daha fazla noktalar değerlendirilir.

>> x = linspace(0, 4*pi, 10);

>> figure; plot(x, sin(x));

>> x = linspace(0, 4*pi, 1000); → Daha iyi görünür.

>> figure; plot(x, sin(x));

* x ve y vektörleri aynı boyutta olmalı yoksa hata çıkarılır.

>> plot([1,2],[1,2,3])

??? Error using plot =>

Vectors must be same lengths.

• Başlık eklemek için

>> title('Plot Title')

- 2 xis etiketleri için

```
>> xlabel('x-label');
```

```
>> ylabel('y-label');
```

ilk grafik oluşturulmalı

```
>> plot(x,y)
```

```
>> title('sinüzph')
```

```
>> xlabel('x')
```

```
>> ylabel('sin(x)')
```

```
>> grid 'on' → ile gridler gösterilmiş olur
```

- Plotun line color, marker stili ve line stili değiştirilebilir.

```
>> plot(x,y,'k.-');
```

color marker style line style

- Plot line style, ihmal edilerek noktalar birbirine bağlandırılabilir.

```
>> plot(x,y,'o')
```

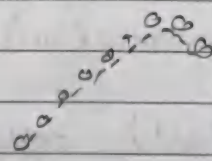
marker's

default color = blue

```
>> plot(x,y,'ro--')
```

circle
(x=y seçisidir)

dash line



şeklinde olur.

k ⇒ black color'dır.

— → simple line

```
>> plot(x, sin(x), 'gd:')
```

green diamond dot line

>> doc plot ile dokumentasyonu görebiliriz.

m → magenta rengi (pembemsi)

ya da rengimizi şöyle değiştirebiliriz:

>> plot(x, y, 'color', [0.5, 0.5, 1.0])

1.0 → üst sınır.

>> plot(x, y, 'linewidth', 2.0)

kalınlığı

Bunları ayrıca pencereden de yapabiliriz. (Açılan figure penceresi)

şef tıklaz yapabiliriz. gizlin olduğu yerde.

Ama üstten fareyi seç ilk başta

Ayrıca property inspector den pozisyon vb. değiştirilebilir.

Show legend ile veri bilgisi grafikte gösterilir.

Save figure ile fig şeklinde veya resim, pdf şeklinde de grafikimizi kaydedilebilir.

Ayrıca export da edilebilir.

Mesela sin.tif şeklinde kaydettiyssek;

>> figure; imshow(sin) yapılmazak zayılabılır

>> legend('sin') şeklinde veri bilgisi eklenebilir.

show code ile graf kodu alınıp sonra kullanılabilir.

ör

```
>> x = -pi:pi/20:pi;  
>> y = tan(sin(x)) - sin(tan(x));  
>> plot(x,y, '--s', 'linewidth', 2, 'Color', [1 0 0],  
        'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'g',  
        'MarkerSize', 10)
```

>> doc Line Properties % Plotting detayları

Cartesian Plots

//Bunu gözden geçirdik detaya girme.

semilogx → logaritmik ölçeklendirme x ekseninde

semilogy → " " " " y "

loglog → x-y ekseninde logaritmik ölçeklendirme

plotyy → 2D line plot: her iki taraf y eksenini alır.

```
>> x = 0:100;
```

```
>> semilogy(x, exp(x), 'k.-') → y ekseninde olur.
```

GRAPHICS

1) Bar Graph

```
>> x = categorical({'ff', 'xx', 'zz'});
```

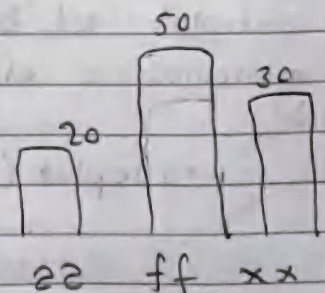
```
>> y = [50, 30, 20];
```

```
>> x = reordercats(x, {'ff', 'xx', 'zz'});
```

```
>> bar(x,y)
```

```
>> title, xlabel vb yazılabilir
```

Burisi sırasıyla
alın

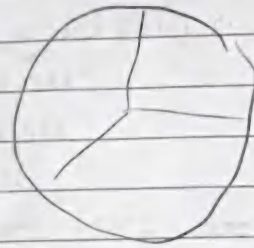


olun

2) Pie Charts

pie, pie3
2D 3D

```
>> x = 100 * rand(1,5);  
>> pie(x)  
>> title('Pie Chart');  
>> legend('FF', 'DC', 'CC', 'BB', 'AA');
```

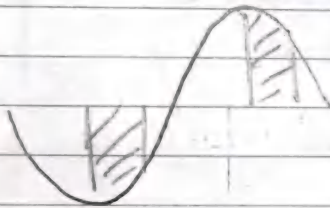


sekinde olur.

3) Area Graph

```
>> x = [-pi:0.01:pi]; y = sin(x);  
>> plot(x,y); hold on;  
>> area(x(200:300), y(200:300));  
>> area(x(500:600), y(500:600)); hold off.
```

yapmayı
unutma.



→ sekinde taralı yerler boynur.

Multiple plots in one Figure window

subplot tek bir figure penceresinde farklı plotları aynı aynı göstermemize olanak sağlar.

- subplot(2,3,1) 2 rows ve 3 column, figure olusturur
1. plot dizim anlemini verir.

`subplot(2,3,4:6)` → bir range ile aktif edilebilir.

```
>> x = linspace(0, 2*pi, 50);  
>> subplot(2,2,1), plot(x, sin(x)), xlabel('x'), ylabel('sin(x)');  
    " 2,2,2), " cos  
    2,2,3 sin(2*x)  
    2,2,4 plot(x, cos(2*x)) -- 'cos(2*x)';
```

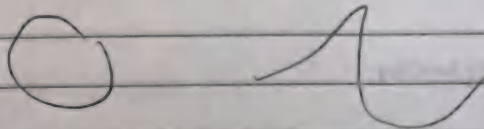
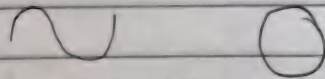
Öb

```
>> x = linspace(0, 2*pi);  
>> subplot(2,2,1); plot(x, sin(x), x, cos(x), '--')  
>> axis([-1 9 -1.5 1.5])  
>> xlabel('x'), ylabel('y'), title('Place (1,1)'), grid on  
>> subplot(2,2,2); plot(exp(i*x)), title('Place (1,2):  
    z = e^{ix}')  
>> axis square, text(0,0, 'i is complex')
```

```
>> subplot(2,2,3);  
>> polar(x, ones(size(x))), title('Place (2,1)')
```

```
>> subplot(2,2,4);  
>> semilogx(x, sin(x), x, cos(x), '--')  
>> title('Place : (2,2)'), grid on  
>> legend('sin', 'cos', 'Location', 'Southwest')
```

hoca isterinde pek durmazdı sizce mi?



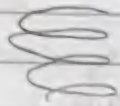
şeklinde grafikler
düşüyor.

3D Plotting

2D gibi 3D plotlar da kolayca yapılabilir.

```
>> t = 0:0.001:4*pi;  
>> x = sin(t);  
>> y = cos(t);  
>> z = t;  
>> plot3(x,y,z,'k','linewidth',2);  
>> zlabel('Time');  
>> grid on;
```

3d yzay



sekinde cıkar.

Some Useful Plotting Commands

Built-in axis modes

- axis square: makes the current axis look like a box.
- axis tight: fits axes to data.
- axis equal: makes x and y scales the same.
- axisxy: puts the origin in bottom left corner (default).
- axisij: puts the origin in the top left corner (default for matrices/images).

Closing figures

- close([1 3]) closes figures 1 ve 3
- close all.

MATLAB'da en yukerda plots sekmesine tiklayarak kolayca plot olusturulabilir.

Workspacedeki deęiskenleri secerek plot secip kolayca olusturulabilir.

errorbar ile sapmalar gosterilir.

```
>> x = linspace(0, 3*pi, 200);  
>> y = cos(x);  
>> y = cos(x) + rand(1, 200);  
günlü ekledik.
```

```
>> scatter(x, y) → sadece markerler var. Veri madenciliği vb  
eğil indirmede  
clustering de kullanılabilir.  
>> c = linspace(1, 10, length(x));  
>> scatter(x, y, [3, c]) → ile renklendirmiş oluyoruz.
```

Visualizing Matrices

★ Herhangi bir matrix resim gibi görselleştirilebilir.

```
>> mat = reshape(1:10000, 100, 100);
```

```
>> imagesc(mat);  
>> colorbar
```

→ 100x100'e
1'den 10000'e
kadar git.

• `imagesc` ⇒ automatically scales the values to span (kapsamak) the entire colormap.

• 'Color axis limiti için

```
>> caxis([3000 7000]) // limit veriliyor
```

veriler renklenmiş olur.

```
>> colormap('gray'); // veya colormap('gray'); gibiler renklendirmeler yapılabilir.
```


Colormap'ler;

jet → renkli

gray → siyah beyaz

cool →

hot(256)

parlak defaultdır.

★ Surf (Gök kullanışlı ve önemli)

- meshgrid(x,y) = function $z = f(x,y)$ fonksiyonun 3D plot'unda z'yi oluşturmak amacıyla; x ve y elementlerini içeren tüm kombinasyonların gridini üretir.

surf ⇒ puts vertices (köşeler) at specified points in space
 $x; y; z$, ve bunları birbirine bağlayarak yüzey oluşturur

x ve y vektörleri;

>> x = -pi: 0.1: pi; 1x63

>> y = -pi: 0.1: pi; 1x63

matris oluşturmak için meshgrid kullanımı

>> [x,y] = meshgrid(x,y); // Tüm gridleri verir.
63x63 olur.

Bu

X y 1.

büyük

x ve y
den farklı

Fonksiyon değerini hesapla

büyük

>> z = sin(x) * cos(y)

sonuç

bu almazsz sonuç değerişyon
element-wise operation

Plot the surface

büyükler

>> surf(X,Y,z)

renkler değerişyon gösterir (hot ⇒ bigger)

Ayrıca colormap kullanılabilir.

Ayrıntılı Detay için \Rightarrow doc surf.

Surface Shading türleri;

\gg shading faceted \rightarrow default
 \gg shading flat
 \gg shading interp } gridleri değiştiriyor (smoother)

$$\sqrt{a^2 + b^2} = r$$
$$r \cdot \cos^2$$

contour; kullanılacak 2 Boyutlu yüzeyler oluşturulabilir.

\gg contour(x, y, z, 'Linewidth', 2)

surf

argümanları

color indicates height

linestyle properties değiştirilebilir.

colormap yazılabilir.

\gg hold on

\gg mesh(x, y, z) \rightarrow detay gösterilir. İstene ekler 3D halini görülebilir.

SORULAR

New Live Script.

1) Plot the following functions (you will need to decide on appropriate ranges for x)

- $y = 1/x$ with blue dashed line and circle markers
- $y = \sin(x) \cdot \cos(x)$ with a red dotted line and diamond markers
- $y = 2x^2 - 3x + 1$ with green solid line and cross markers.
- A circle of radius = 5

Turn the grid on in all your plots. Remember to label axes and add a title.

NOT

>> A = [1, 2; 3, 4]

A =
1 2
3 4

>> b = [-1 0; -1 2]

b =
-1 0
-1 2

>> A * b → Matris çarpımıdır

ans =
-3 4
-7 8 olur.

>> A .+ b → element-wise operation
elementleri
tek tek olarak
çarpılır

ans =
-1 0
-3 8

olur.

1. cınnı cevabı;

>> x = -10:1:10;

>> y = 1./x;

>> plot(x, y, 'bo--'), title('y=1/x'), xlabel('x'),
ylabel('y');

>> grid on;

bu işe
gelebilir.

2. cınnı cevabı; → lineer aralıklı olarak

>> x = linspace(0, 2*pi, 50);

>> y = sin(x) .* cos(x); // matris çarpımı olsaydı k x m m x n

k x n olur.

>> plot(x, y, 'rd:'), grid on, label --- title --- olur.

Transpozu olsaydı birinin noktasız
çarpılabirdi.

3. cümlün cevabı;

>> x = -10:1:10;

>> y = 2 * x.^2 - 3 * x + 1

skaler eleman olduğu için noktaya gerek yok.

>> plot(x, y, 'gx-'), grid on --- diye olur.

4. cümlün cevabı;

$$x^2 + y^2 = 25$$

>> r = 5;

>> theta = linspace(0, 2 * pi, 100);

>> x = r * cos(theta);

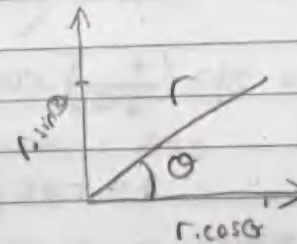
>> y = r * sin(theta);

>> plot(x, y, 'k', 'linewidth', 2), grid on, x12 ---

>> axis equal

// Düzgünlemek için

// Kenlerden sıkıktırmak için axis tight;



NOT

pi = 3,14 mi 180° mi

Çemberin çevresinin 360°'te 1'ini gören merkez açı 1°'dir.

Radyan, bir dairede yarıçap uzunluğundaki yay parçasını gören merkez açıya eşit açı ölçme birimidir.

1 radyan $\frac{180}{\pi}$ ye de yaklaşık 57.2958 derecedir.

$$\frac{\pi}{3} \text{ radyan} = \frac{\pi}{3} \cdot \frac{180}{\pi} = 60^\circ \text{ olur.}$$

$$2\pi \cdot \frac{180}{\pi} = 360^\circ \text{ olur.}$$

ör

Sphere ıgı dolu dıire cızmek ıgın

```
thetaz = linspace(0, 2*pi, 360);
```

```
phi = linspace(0, 2*pi, 360);
```

```
[th, ph] = meshgrid(thetaz, phi);
```

```
R = ones(size(th));
```

```
x = R.*sin(th). *cos(ph);
```

```
y = R.*sin(th). *sin(ph);
```

```
z = R.*cos(th);
```

```
surf(x, y, z);
```

ör

Plot the following 3D curve using the plot3 function.

$$x = \sin\left(\frac{t}{2c}\right) \cdot \cos(t) \quad y = \sin\left(\frac{t}{2c}\right) \cdot \sin(t)$$
$$z = \cos\left(\frac{t}{2c}\right) \quad c=5 \quad 0 \leq t \leq 10\pi$$

```
>> c=5;
```

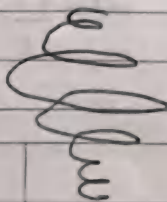
```
>> t = linspace(0, 10*pi, 200);
```

```
>> x = sin(t./(2*c)). *cos(t);
```

```
>> y = sin(t./(2*c)). *sin(t);
```

```
>> z = cos(t./(2*c));
```

```
>> plot3(x, y, z), xlabel('x'), ylabel('y'), zlabel('z'), grid on,  
title('Helix');
```



→ sekli cıkar.

surf için meshgrid 2D veya vektör olmalı.
surf 'de boyutlar uyumlu olmalı.

BÖLÜM 4: Linear Algebra with MATLAB

MATLAB'de lineer cebir fonksiyonları hızlı, sayısal işlemler için kullanılır.

matrices in the matlab dokümanı incelenebilir, } mathworks.com
ve
Linear Algebra in MATLAB bakmak faydalı olacaktır.

Matrices

- A matrix is a rectangular array of numbers, symbols or expressions arranged in rows and columns.

- an individual entry of a matrix is an element: a_{ij}

- The size of matrix is given as m rows by n columns, or simply m by n ($m \times n$)

$$A = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & & a_{2n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix} m \times n$$

- $1 \times n$ matrisleri \rightarrow row vectors

- $m \times 1$ matrisleri \rightarrow column vectors olarak adlandırılır.

Matrix Dimensions

• MATLAB has several functions to determine the number of elements in and the dimensions of variables:

- `numel(vec)` returns the number of elements in the vector.
- `numel(mat)` returns the total number of elements in the matrix (the product of the number of rows and the number of columns).
- `length(vec)` returns the number of elements in the vector.
- `length(mat)` returns either the number of rows or the number of columns in the matrix, whichever is larger.
- `size(mat)` returns the number of rows and the number of columns of the matrix.
- `size(vec)` returns the number of rows and the number of columns of the vector.

Matrix Operations

- `isequal(A,B)`: two matrices are considered equal if they have the same dimensions, and all corresponding elements equal.
- $A+B$, $A-B$: matrix toplema ve cikarma performed by adding or subtracting the corresponding elements. Two matrices must be the same size.
- $A * c$: Scalar matrix multiplication is performed by multiplying each element by the same scalar.
- `diag(A)`: produces a column vector containing the elements of the main diagonal of A

```
>> A = magic(3)
```

```
A =
```

```
8 1 6
3 5 7
4 9 2
```

```
>> v = diag(A)
```

```
v =
```

```
8
```

```
5
```

```
2
```

• $\text{diag}(v)$: produces a square

diagonal matrix with the elements of vector v on the main diagonal.

```
>> B = diag(v)
```

```
B =
```

```
8 0 0
```

```
0 5 0
```

```
0 0 2
```

Matrix Vector Multiplication

• If A is an $m \times n$ matrix, and x is an $n \times 1$ vector, then the "product of Ax " is a $m \times 1$ vector.

```
>> A = [1 0 -1; 2 -3 2];
```

```
>> x = [2 1 0]';
```

```
>> A*x
```

```
ans =
```

```
2
```

```
1
```

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & -3 & 2 \end{bmatrix}_{2 \times 3} \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}_{3 \times 1} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}_{2 \times 1}$$

Matrix Multiplication

• If A is $m \times n$ matrix and B is an $n \times p$ matrix, their matrix product AB is an $m \times p$ matrix, in which the m entries across a row of A are multiplied with the n entries down a column of B and summed to produce an entry of AB .

```
>> A = [1:3; 4:6]
```

```
A =
```

```
1 2 3
```

```
4 5 6
```


$$\gg B = \begin{bmatrix} -1 & 1 \\ -2 & 2 \\ -1 & 1 \end{bmatrix}$$

$$B =$$

$$\begin{bmatrix} -1 & 1 \\ -2 & 2 \\ -1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 1 \\ -2 & 2 \\ -1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 1 \\ -2 & 2 \\ -1 & 1 \end{bmatrix}$$

$$3 \times 2$$

$$A = 2 \times 3$$

$$\gg C = A * B$$

$$C =$$

$$\begin{bmatrix} -8 & 8 \\ -20 & 20 \end{bmatrix}$$

$$\begin{bmatrix} -8 & 8 \\ -20 & 20 \end{bmatrix}$$

$$A_{m \times n} B_{n \times p} = [C]_{m \times p}$$

$$C_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$$

Vector Norms

- The euclidean norm or 2-norm of a vector v with n elements is defined by

$$\|v\| = \sqrt{\sum_{k=1}^n |v_k|^2}$$

$$\gg v = \begin{bmatrix} -2 & 3 & -1 \end{bmatrix};$$

$$\gg \text{norm}(v)$$

$$\text{ans} =$$

$$3.7417$$

- The general definition for the p norm of a vector v that has n elements is

$$\|v\|_p = \left(\sum_{k=1}^n |v_k|^p \right)^{1/p}$$

$p \rightarrow$ any positive real value, Inf or $-\text{Inf} \rightarrow \text{Sonsuz}$.

Determinant

mesela $A = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ $B = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$ $C = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ $D = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

olsun

A, B, C, D yi $\begin{pmatrix} 3 & 0 \\ 1 & 1 \end{pmatrix}$ ile her elemanını çarparsak altında
0 ise elemanı 0 olur. değilse veya noktasızdır.

ABCD'nin elemanı bunun determinantiyle çarpılırsa yeniden eleman olur

Matris tersi, lineer işlemler vb. için kullanılabilir. |A| ile gösterilir.
2x2 matris için;

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad |A| = ad - bc$$

3x3 matris için;

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \quad |A| = a(ei - fh) + b(fg - di) + c(dh - eg)$$

şeklinde olur.

- Geometrik olarak yorumlarsak; determinant, area(2D), volume(3D) veya hypervolume(4D or higher) nin nasıl olduğunu X nokta kümesinin değişimine göre when matrix A is applied to X.

Sinavda matlab'de yapılması muhtemel,
Bir problemten grafik çıkarma vb olabilir. Sorular özellecek.

Determinant: bir şeklin alanının ne kadar değişeceğini gösterir. 3D şeklin hacminin, 4D hipervolume'nin ne kadar değişeceğini gösterir.

matlab'de $\det(\text{matris}) \rightarrow$ seklinde olur.

Matrix Inverse

Bir matrisin tersi, bu matris ile çarpıldığında birim matrisi elde ederiz.

$$A \cdot A^{-1} = I$$

>> $A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$

A =

1 3
2 4

* >> $\text{inv}(A)$

ans =

-2.0000 1.5000
1.0000 -0.5000

* >> $A * \text{inv}(A)$

ans =

1 0
0 1

Extra NOT

mathisfun.com \rightarrow güzel site.

Inverse of a matrix using elementary row operations.
(Also called Gauss-Jordan method)

$$\begin{array}{c} [A | I] \\ \downarrow \downarrow \\ [I | A^{-1}] \end{array}$$

Play around with the rows (adding, multiplying or swapping) until we make matrix A into the Identity matrix.

And also doing the changes to an Identity matrix it magically turns into the inverse

ör

$$A = \begin{bmatrix} 3 & 0 & 2 \\ 2 & 0 & -2 \\ 0 & 1 & 1 \end{bmatrix}$$

we start with the matrix A, and write it down with an Identity matrix I next to it:

$$\left[\begin{array}{ccc|ccc} 3 & 0 & 2 & 1 & 0 & 0 \\ 2 & 0 & -2 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array} \right]$$

→ Augmented Matrix olarak adlandırılır.

Identity matrix, köşeleri 1 diğerleri 0, I ile gösterilir.

1) 2. satırı 1. satıra ekle

$$\left[\begin{array}{ccc|ccc} 5 & 0 & 0 & 1 & 1 & 0 \\ 2 & 0 & -2 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array} \right]$$

2) 1. satırı 5'e böl

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0.2 & 0.2 & 0 \\ 2 & 0 & -2 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array} \right]$$

Solving Linear Equation Systems

doc mldivide

Bir lineer denklemin çözümü bulunabilir.

$$\begin{aligned}x_1 + 3x_2 &= 1 \\ 2x_1 + 7x_2 &= 3\end{aligned}$$

$$>> A = [1 \ 3; 2 \ 7];$$

$$>> b = [1 \ 3]';$$

$$>> x = A \setminus b$$

$$x = \begin{matrix} -2 \\ 1 \end{matrix}$$

oluyor

1

x_1 ve x_2 oluyor.

Vector Operations

Dot Product (Nokta Çarpımı)

Vektörlerin çarpımı, karşılıklı elementlerinin çarpımının toplamına eşittir.

$$>> u = [1, -1];$$

$$>> v = [2, 3];$$

$$\star >> \text{dot}(u, v);$$

$$\text{ans} =$$

$$-1$$

Vektörler arasındaki açıyı bulabiliriz.

$$x \cdot y = \|x\| \cdot \|y\| \cdot \cos \theta$$

$$\cos \theta = \frac{x \cdot y}{\|x\| \cdot \|y\|}$$

olur.

$$\star x = \text{dot}(u, v) / (\text{norm}(u) * \text{norm}(v));$$

$$2 \cos(x)$$

$$\text{ans} = 1.7682$$

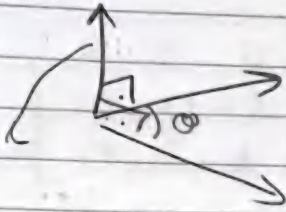
θ olur.

$$\begin{matrix} u = 3 & 4 \\ v = 3 & 4 \end{matrix} \left. \vphantom{\begin{matrix} u \\ v \end{matrix}} \right\} \text{olursa } \theta = 0 \text{ olur. } \cos 0 = 1$$

$$5.5$$

$$5.5$$

Cross Product (vektörel çarpım)



İki vektörün vektörel çarpımından
dik yeni bir vektör.

$$\gg A = \begin{bmatrix} 4 & -2 & 1 \end{bmatrix}$$

$$\gg B = \begin{bmatrix} 1 & -1 & 3 \end{bmatrix}$$

$$\gg \text{cross}(A, B)$$

$$\text{ans} = 1 \times 3$$

$$\begin{bmatrix} -5 & -11 & -2 \end{bmatrix}$$

Aslında;

$$\begin{vmatrix} i & j & k \\ 4 & -2 & 1 \\ 1 & -1 & 3 \end{vmatrix}$$

determinantı olur.

i, j, k'nin katsayıları

Eigenvalues and Eigenvectors

- A n*n matris olsun. x , \mathbb{R}^n 'den non-zero bir vektör ve λ herhangi bir scalar (zero olabilir) öyleyse;

$$Ax = \lambda x$$

- we then call x an eigenvector of A ve λ an eigenvalue of A .

- we will often call x the eigenvector corresponding to or associated with λ and we will often call λ the eigenvalue corresponding to or associated with x .

- Matlab has a built-in function to find eigenvalues and eigenvectors of a given matrix: eig.

- $[V, D] = \text{eig}(A)$: returns diagonal matrix D of eigenvalues and matrix V whose columns are the corresponding right eigenvectors.

ör

$$\gg A = \begin{bmatrix} -4 & 2 \\ 3 & -5 \end{bmatrix};$$

$$\gg [V, D] = \text{eig}(A);$$

$V =$

$$\begin{pmatrix} 0.7071 \\ 0.7071 \end{pmatrix}$$

ilk vektör

$$\begin{pmatrix} -0.5547 \\ 0.8321 \end{pmatrix}$$

ikinci vektör

$$D = \begin{pmatrix} -2 & 0 \\ 0 & -7 \end{pmatrix}$$

eigen value of matrix A

bunlar ilişkili

$Ax = \lambda x$ bunlar beklenizle yazılabilir.

$$\gg v = V(:, 1)$$

$$v = \begin{pmatrix} 0.7071 \\ 0.7071 \end{pmatrix}$$

ikinci vektör için
 $v = V(:, 2)$ yapılır
 $D(2, 2)$

$$\gg \text{lambd}_2 = D(1, 1);$$

$$\star \gg A * v$$

ans =

$$\begin{pmatrix} -1.4142 \\ -1.4142 \end{pmatrix}$$

$$A * v = \text{lambd}_2 * v \text{ oldu}$$

$$\star \gg \text{lambd}_2 * v$$

$$\text{ans} = \begin{pmatrix} -1.4142 \\ -1.4142 \end{pmatrix}$$

Türkçede özdeğer ve özvektör deniyor.

x = özvektör
 λ = özdeğer

2 boyutlu bir düzlem olsun.

35

* A matrisiyle çarpıldığında aynı doğrultuya sahip olan vektörler özvektörlerdir.

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} x \xrightarrow{v} (1,0) \quad \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

bu vektör
öz vektör
olur.

λ ile aynı
doğrultuda
h

özdeğer ise 2 kat büyüdüğü için 2 olur.

$$2 \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Bir özdeğere karşılık bir tane özvektör gelebilir.

$$Ax = \lambda x \Rightarrow (A - \lambda I)x = 0$$

$\det(A - \lambda I) = 0$ sağlayan λ değerleri özdeğerlerdir.

$$\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} = \begin{bmatrix} 2-\lambda & 0 \\ 0 & 1-\lambda \end{bmatrix} = 0$$

Lineer Dönüşümler (Linear Transformations)

n boyutlu reel sayılar uzayında m boyutlu reel sayılar uzayına bir vektör vb. dönüştüren anz kâdiren yeni bir vektör bulan dönüşümlerdir. Lineer olması için şartlar var.

$$T(x, y) = (x^2, x-y, y^2)$$

$$T(1, 2) = (1, -1, 4) \quad \times \text{ Bu lineer dönüşüm mî.}$$

Şartlar;

1) $T(u+v) = T(u) + T(v)$ for all u, v ; in the domain of T .

2) $T(cu) = c T(u)$ for all scalars c and all u in the domain of T .

ör

$$T(x, y) = (x-y, x+y) \text{ olsun.}$$

$$u(u_1, u_2) \quad v(v_1, v_2) \text{ ise;}$$

$$\begin{array}{ccc} \uparrow & & \uparrow \\ x' \text{ in yerine} & & y' \text{ nin yerine} \end{array}$$

$$= (u_1 + v_1 - u_2 - v_2, u_1 + v_1 + u_2 + v_2)$$

$$= (u_1 - u_2, u_1 + u_2) + (v_1 - v_2, v_1 + v_2)$$

$$= (x-y, x+y) \Rightarrow T(u) + T(v) \text{ oldu şğıladı.}$$

$T(x-y, x+y)$ bu dönüşüme karşılık gelen matris; katsayılarıdır.

$$\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

$$T(1, 4) = (3, 5)$$

olur yâdî

← böyle hesaplanır.

$$\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} = \begin{bmatrix} -3 \\ 5 \end{bmatrix}$$

extra not;

Lineer Dönüşüm

Lineer dönüşümler, mühendislik ve fizik sahalarında sıkça kullanılan ve matrisler ile ifade edilebilmesinden dolayı kullanımı oldukça pratik fonksiyonlardır. Özellikle hareket ifade eden lineer dönüşümler; bilgisayar grafikleri, makine hareketleri, animasyon ve robot teknolojisinde sıkça kullanılmaktadır.

Lineer dönüşümler, \mathbb{R}^n uzayındaki elemanları, noktaları, vektörleri \mathbb{R}^m uzayındaki elemanlara, noktalara, vektörlere dönüştüren özel fonksiyonlardır:

$$f: \mathbb{R}^n \rightarrow \mathbb{R}^m$$

Tanım: $T: \mathbb{R}^n \rightarrow \mathbb{R}^m$ dönüşümü, her $\vec{u}, \vec{v} \in \mathbb{R}^n$ ve $c \in \mathbb{R}$ için

$$T(\vec{u} + \vec{v}) = T(\vec{u}) + T(\vec{v})$$

$$T(c\vec{u}) = cT(\vec{u})$$

koşullarını sağlıyorsa \mathbb{R}^n 'den \mathbb{R}^m 'ye bir lineer dönüşüm denir.

Ör

$$T: \mathbb{R}^3 \rightarrow \mathbb{R} \rightarrow T(x_1, x_2, x_3) = (x_1 * x_2 * x_3)$$

→ tek boyutlu

$$T: \mathbb{R}^2 \rightarrow \mathbb{R}^3 \rightarrow T(x_1, x_2) = (x_1 + x_2, x_1 - x_2, x_1 * x_2)$$

ör

$$T(x, y) = (x+y, 2x-y, y)$$

$T: \mathbb{R}^2 \rightarrow \mathbb{R}^3$ tanımlı fonksiyonun doğrusal dönüşüm olduğunu gösteriniz.

2 şartı da sağlamalıdır.

$$T(u+v) = T(u) + T(v)$$

$$\vec{u} = (x_1, y_1)$$

$$\vec{v} = (x_2, y_2) \text{ olsun}$$

$$T(\vec{u} + \vec{v}) = T(\vec{u}) + T(\vec{v})$$

$$\vec{u} + \vec{v} = (x_1 + x_2, y_1 + y_2)$$

$$T(x_1 + x_2, y_1 + y_2) = (x_1 + x_2 + y_1 + y_2, 2x_1 + 2x_2 - y_1 - y_2, y_1 + y_2)$$

$$T(\vec{u}) = T(x_1, y_1) = (x_1 + y_1, 2x_1 - y_1, y_1)$$

$$T(\vec{v}) = T(x_2, y_2) = (x_2 + y_2, 2x_2 - y_2, y_2)$$

$$(x_1 + x_2 + y_1 + y_2, 2x_1 + 2x_2 - y_1 - y_2, y_1 + y_2) = (x_1 + y_1, 2x_1 - y_1, y_1) + (x_2 + y_2, 2x_2 - y_2, y_2)$$

$$T(\vec{u} + \vec{v})$$

$$T(\vec{u}) + T(\vec{v})$$

$$= (x_1 + x_2 + y_1 + y_2, 2x_1 + 2x_2 - y_1 - y_2, y_1 + y_2)$$

olur.

1. şart sağlandı.

$$T(x, y) = (x+y, 2x-y, y)$$

2. soru

$$c \in \mathbb{R} \quad u = (x_1, y_1) \text{ olsun} \quad T(c \cdot u) = c \cdot T(u) \text{ olmalı.}$$

$$c \cdot u = (c \cdot x_1, c \cdot y_1)$$

$$T(c \cdot u) = (cx_1 + cy_1, 2cx_1 - cy_1, cy_1)$$

$$T(u) = (x_1 + y_1, 2x_1 - y_1, y_1)$$

$$c \cdot T(u) = (c \cdot x_1 + c \cdot y_1, 2c \cdot x_1 - c \cdot y_1, c \cdot y_1)$$

$$T(c \cdot u) = c \cdot T(u) \text{ olur.}$$

Bu yüzden $T(x, y) = (x+y, 2x-y, y)$ lineer dönüşüm fonksiyonudur.

ÖR

Aşağıdaki dönüşümlerin lineer dönüşüm olup olmadığını araştırın.

i) $T: \mathbb{R}^3 \rightarrow \mathbb{R}^2, T(x, y, z) = (x+y, x+z)$

ii) $T: \mathbb{R}^3 \rightarrow \mathbb{R}^3, T(x, y, z) = (x, 2y, z)$

iii) $T: \mathbb{R}^3 \rightarrow \mathbb{R}^2, T(x, y, z) = (y^2 + z^2, 2xy)$

iv) $T: \mathbb{R}^3 \rightarrow \mathbb{R}^3, T(x, y, z) = (1, 1, 1)$

v) $T: \mathbb{R}^3 \rightarrow \mathbb{R}^3, T(x, y, z) = (0, 0, 0)$

Gör

i)

$$\vec{u} = (u_1, u_2, u_3), \vec{v} = (v_1, v_2, v_3) \in \mathbb{R}^3$$

ve

$$c \in \mathbb{R} \text{ için } T(\vec{u} + \vec{v}) = T(\vec{u}) + T(\vec{v}) \text{ ve}$$

$$T(c\vec{u}) = c T(\vec{u}) \text{ olduğunu göstermeliyiz.}$$

$$T(\vec{u} + \vec{v}) = (u_1 + v_1 + u_2 + v_2, u_1 + v_1 + u_3 + v_3)$$

$$T(\vec{u}) = (u_1 + u_2, u_1 + u_3) \quad T(\vec{v}) = (v_1 + v_2, v_1 + v_3)$$

$$= T(\vec{u}) + T(\vec{v}) \quad 1. \text{ şart sağlanır.}$$

$$T(c\vec{u}) = c \cdot T(\vec{u})$$

$$u = (u_1, u_2, u_3) \text{ olsun.}$$

$$T(cu_1, cu_2, cu_3) = c \cdot T(u_1, u_2, u_3)$$

$$(cu_1 + cu_2, cu_1 + cu_3) = c \cdot (u_1 + u_2, u_1 + u_3)$$

$$= c \cdot T(\vec{u}) \quad 2. \text{ şart sağlanır.}$$

T lineer dönüşümdür.

ii)

$$\vec{u} = (u_1, u_2, u_3)$$

$$\vec{v} = (v_1, v_2, v_3)$$

$$T(\vec{u} + \vec{v}) = (u_1 + v_1, 2(u_2 + v_2), 1)$$

$$T(\vec{u}) = (u_1, 2u_2, 1) \quad T(\vec{v}) = (v_1, 2v_2, 1)$$

$$T(\vec{u}) + T(\vec{v}) = (u_1 + v_1, 2u_2 + 2v_2, 2) \quad \times \text{ eşit değil}$$

birinci şart sağlanmaz. T lineer dönüşüm değil.

iii)

$$\vec{u} = (u_1, u_2, u_3) \quad \vec{v} = (v_1, v_2, v_3)$$

$$T(\vec{u} + \vec{v}) = ((u_2 + v_2)^2 + (u_3 + v_3)^2, 2(u_1 + v_1)(u_2 + v_2)) \text{ olur.}$$

hace buna bakmamız

$$T(c\vec{u}) = (c^2 u_2^2 + c^2 u_3^2, 2c^2 u_1 u_2)$$

$$c \cdot T(\vec{u}) = c(u_2^2 + u_3^2, 2u_1 u_2) \quad \rightarrow \text{Bu } B_{u_1, u_2} \quad c^2 \cdot T(\vec{u}) \text{ olur.}$$

esit olmaz

birinci şartta da lineer olmadığı görülür.

iv)

$$T(\vec{u} + \vec{v}) = T(u_1 + v_1, u_2 + v_2, u_3 + v_3) = (1, 1, 1)$$

$$T(\vec{u}) = T(u_1, u_2, u_3) = (1, 1, 1) \text{ Ne koyarsak sabit duca}$$

$$T(\vec{v}) = T(v_1, v_2, v_3) = (1, 1, 1)$$

$$(2, 2, 2)$$

Lineer dönüşüm değildir.

v) $T: \mathbb{R}^3 \rightarrow \mathbb{R}^3$, $T(x, y, z) = (0, 0, 0)$ dönüşümü O dönüşümüdür. \mathbb{R}^n 'de tanımlı her \vec{x} vektörünü \mathbb{R}^m 'de O vektörü ile eşler ve lineer bir dönüşümdür. Lineer

Teorem

Lineer Dönüşümün Matris Gösterimi

$T_A: \mathbb{R}^n \rightarrow \mathbb{R}^m$ bir lineer dönüşüm olmak üzere, her $\vec{x} = (x_1, \dots, x_n)$ vektörü için

$$T_A(x_1, x_2, \dots, x_n) = (y_1, y_2, \dots, y_m) \text{ ve}$$

$$y_1 = a_{11}x_1 + \dots + a_{1n}x_n$$

$$y_m = a_{m1}x_1 + \dots + a_{mn}x_n$$

esitliklerini sağlayan $m \times n$ mertebeden

$$\text{bir } A = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix} \text{ matrisi vardır.}$$

Buna göre, $m \times n$ mertebeden A matrisine,

$T_A: \mathbb{R}^n \rightarrow \mathbb{R}^m$ dönüşümüne karşılık gelen matris denir ve

$$T_A(\vec{x}) = A\vec{x} \text{ ile gösterilir.}$$

Ör

$T_A: \mathbb{R}^2 \rightarrow \mathbb{R}^3$ ve $\vec{x} = (x_1, x_2) \in \mathbb{R}^2$ olmak üzere

$T(\vec{x}) = (x_1 + x_2, x_2 - x_1, 2x_1 + x_2)$ lineer dönüşümüne karşılık gelen matrisi bulun. $T(1, 4)$ değerini bulduğunuz matrisi kullanarak hesaplayın.

$T(\vec{x}) = (y_1, y_2, y_3)$ şeklinde ifade ederseniz:

$$y_1 = x_1 + x_2$$

$$y_2 = -x_1 + x_2$$

$$y_3 = 2x_1 + x_2$$

esitliklerini yazabiliriz. Elde ettiğimiz lineer denklem sistemini matris formunda yazarsak:

$$T(\vec{x}) = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \Rightarrow \vec{y} = A \cdot \vec{x}$$

elde edilir.

Buna göre $T(1,4)$ dönüşümünü hesaplırsak;

$$T(1,4) = \begin{bmatrix} 1 & 1 \\ -1 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} = \begin{bmatrix} 5 \\ 3 \\ 6 \end{bmatrix}$$

$T(1,4) = (5, 3, 6)$ olarak bulunur.

Tabi ilk basta lineer dönüşüm olup olmadığına bakılmalıdır.

Dik İzdüşüm Dönüşümü (Projection)

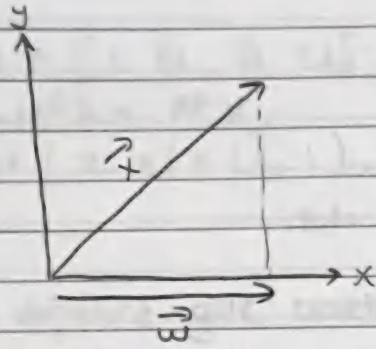
\mathbb{R}^2 uzayında verilen bir noktanın yada konum vektörünün, x eksenine dik izdüşümünü veren dönüşüm ve bu dönüşümün matrisi:

$$I_{2dx} : \mathbb{R}^2 \rightarrow \mathbb{R}^2, I_{2dx}(x, y) = (x, 0) \Rightarrow A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

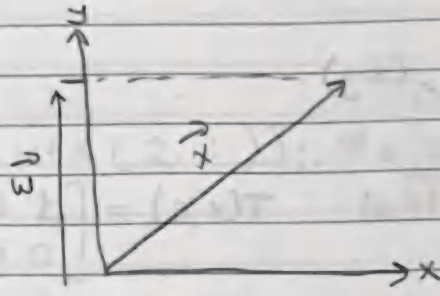
y eksenine dik izdüşümünü veren dönüşüm ve bu dönüşümün matrisi:

$$I_{2dy} : \mathbb{R}^2 \rightarrow \mathbb{R}^2, I_{2dy}(x, y) = (0, y) \Rightarrow A = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

olarak verilebilir.



x eksenine
dik izdüşüm



y eksenine
dik izdüşüm

\mathbb{R}^3 de x eksenine izdüşüm;

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

y eksenine \Rightarrow

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

z eksenine \Rightarrow

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

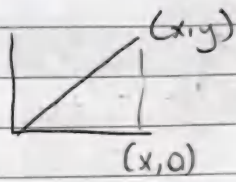
MATLAB deyim

Suppose we have a set of coordinates, and we want to transform it linearly in some fashion with matrix A. Firstly, download the datasets 'logox.txt' ve 'logoy.txt' and put them into your current MATLAB folder.

```
x = [ csvread('logox.txt'); csvread('logoy.txt') ];
```

```
figure; plot(x(1,:), x(2,:), 'k', 'linewidth', 3); title('original  
logo');
```

You should know how to perform the most used transformations, such as rotations, reflections, scaling, shearing, and projects.



$$T(x, 0) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \text{ olur.}$$

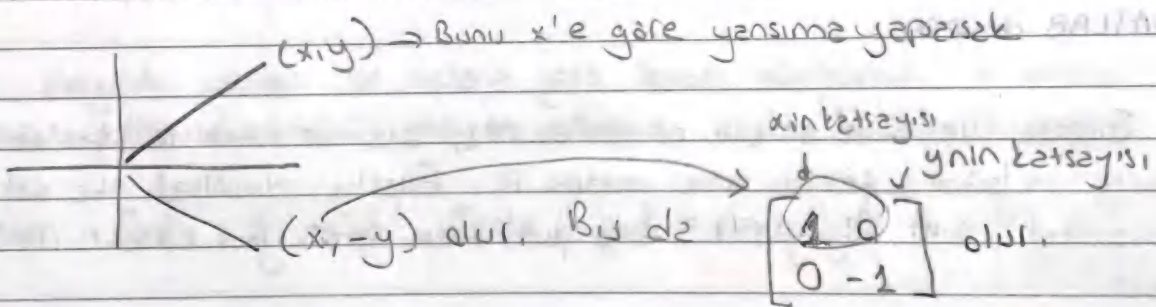
$$>> A = [1, 0; 0, 0];$$

$$>> Y_{\text{-prx}} = A * X; \quad \text{) seklı x eksenine izdüşümü alın.}$$

$$>> \text{figure; plot}(Y_{\text{-prx}}(1, :), Y_{\text{-prx}}(2, :), 'k' \text{---}); \text{---}$$

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \text{ y'ye izdüşüm alınabilir}$$

Reflection'da

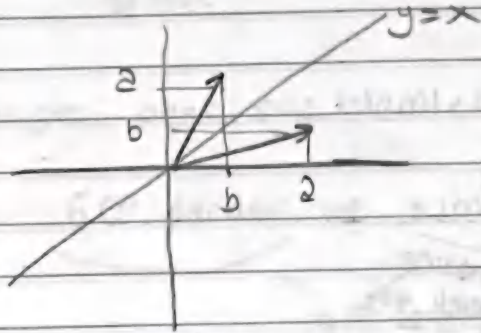


y eksenine göre olursa $(-x, y)$

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \text{ olur.}$$


```
>> A = [1 0; 0 -1];
>> y_ref x = A * x;
>> plot (y_ref x(1, :), y_ref x(2, :)); % x'e göre reflection
```

$y=x$ doğrusuna göre simetliği için



$T(y, x)$ olur.

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

gibi olur.

→ bunu karşarsak $y=x$ 'e göre simetliğini almış oluruz.

orijine göre simetliği ise $T(-x, -y)$ olur

$$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

```
>> A = [0 1; 1 0];
```

```
>> y_ref yx = A * x;
```

```
>> figure; plot (y_ref yx(1, :), y_ref yx(2, :));
```

Orijine göre için $>> A = [-1 0; 0 -1];$

Rotation

Belli açıya göre döndürme yapmak için.

θ açısı için

$$A = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

```

>> theta = 45;
>> A = [cos(theta) -1*sin(theta); sin(theta) cos(theta)];
>> Y_rot = A * x;
>> figure; plot(Y_rot(1, :), Y_rot(2, :));

```

% 45° yönünün teesinde 45° derece döndürür.

Scaling (Ölçekleme)

```

A = [2 0; 0 2]
>> sc = 3;
>> A = [sc 0; 0 sc]
>> Y_sc = A * x
>> plot(Y_sc(1, :), Y_sc(2, :));

```

Aynı olması gerek yok
x ekseninde 2 kat
y'de 1/2 kat 'da yapılabilir

Shear (Yazmaltma)

Belli x ve y'ye göre eğme

Yatay olarak eğmek için;

$$A = \begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix}$$

dikey olarak eğmek için;

$$A = \begin{bmatrix} 1 & 0 \\ b & 1 \end{bmatrix}$$

```

>> b = 2;
>> A = [1 b; 0 1] % yatay eğim,
>> Y_shy = A * x;
>> plot(Y_shy(1, :), Y_shy(2, :));

```


x'e göre yansıma ile y'ye göre yansıma matrislerinin
çarpımı; orijine göre yansımayı verir.

$$(T_A \circ T_B) x = (AB) x = A(B(x))$$

Birleste

Yeni pesi pesi sırası işlemler yapılabilir.

60° döndür ve ^{g'e} simetrisini al.

İkisi çarpılır. Bunu yapmak için.

NOT

Dik izdüşüm Dönüşümü

\mathbb{R}^2 uzayında x eksenine izdüşüm

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

" " y " " " "

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

\mathbb{R}^3 " x eksenine " "

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

\mathbb{R}^3 " y " " " "

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

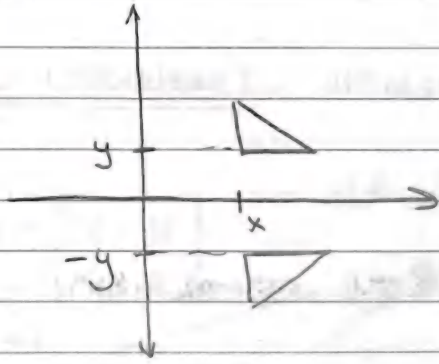
\mathbb{R}^3 " xy düzlemi " "

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

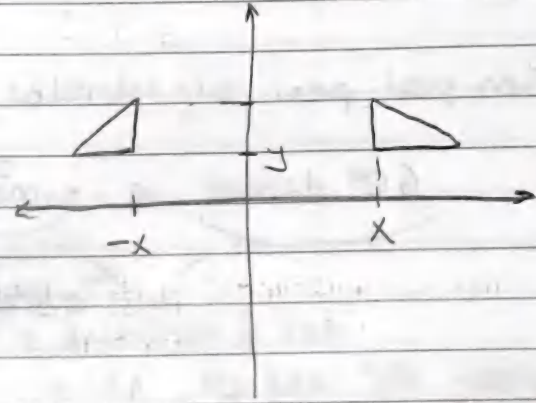
$\begin{pmatrix} x^2 \\ y^2 \end{pmatrix}$ de aynı şekilde olur.

Yansıma - Simetri Dönüşümü

Düzlemde verilen herhangi bir noktanın, bir doğruya göre simetriği olan noktaya yansıma noktası, verilen bir noktanın bu doğruya göre simetriği veren lineer dönüşüme doğruya göre yansıma dönüşümü (reflection transformation), bu yansıma dönüşümüne karşılık gelen matrise de yansıma matrisi denir.



x eksenine göre yansıma



y eksenine göre yansıma

\mathbb{R}^2 'de;

x eksenine göre $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

y eksenine göre $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$

$y=x$ eksenine göre $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

$y=-x$ eksenine göre $\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$

orjine göre $\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$

\mathbb{R}^3 'te;

xy düzlemine göre

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

xz düzlemine

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

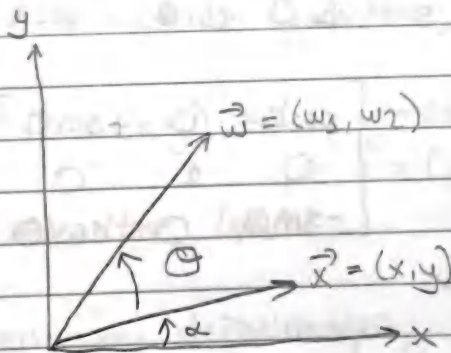
yz düzlemine göre

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Dönme Dönüşümü

Düzlemde verilen bir $P(x,y)$ noktasının, orjin etrafında döndürülmesi bir lineer dönüşümdür. Bu lineer dönüşüme, dönme dönüşümü (rotasyon transformation), bu dönüşüme karşılık gelen matrise de dönme matrisi denir. Herhangi bir $P(x,y)$ noktasını orjin etrafında, saat yönünün tersine Θ açısı kadar döndüren dönüşümü:

$R_\Theta : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ ile gösterelim.



\mathbb{R}^2 uzayını ele alalım. Verilen şekilde her 2 vektörün boyutunu r olduğunu varsayarsak,

$$x = r \cdot \cos \alpha \quad y = r \cdot \sin \alpha \quad \text{ve} \quad w_1 = r \cdot \cos(\alpha + \Theta) \quad w_2 = r \cdot \sin(\alpha + \Theta)$$

$$w_1 = \underbrace{(r \cdot \cos \alpha)}_x \cdot \cos \Theta - \underbrace{(r \cdot \sin \alpha)}_y \cdot \sin \Theta$$

$$w_2 = \underbrace{(r \cdot \cos \alpha)}_x \cdot \sin \Theta + \underbrace{(r \cdot \sin \alpha)}_y \cdot \cos \Theta$$

$$w_1 = x \cdot \cos \theta - y \sin \theta$$

$$w_2 = x \cdot \sin \theta + y \cdot \cos \theta$$

buna göre \mathbb{R}^2 uzayında dönmeye karşılık gelen matris;

$$A = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \text{ olarak elde edilir.}$$

Yani;

\mathbb{R}^2 de; saat yönüne θ açısı için $\begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$

\mathbb{R}^3 'de;

x eksenine göre θ (saat tersi) = $\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}$

y eksenine göre θ (saat tersi) = $\begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}$

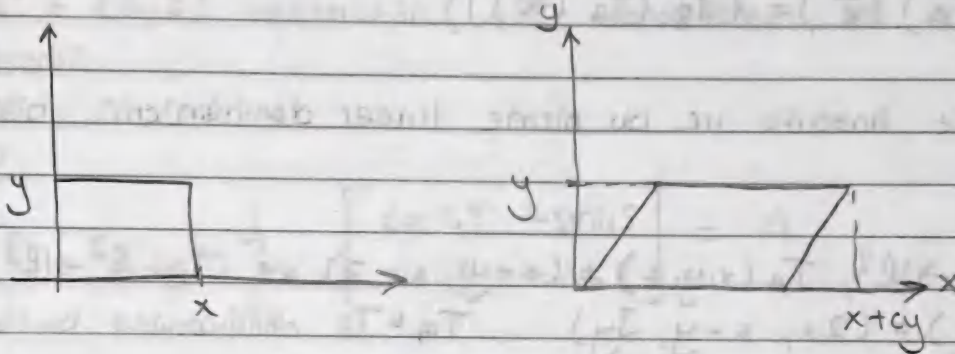
z eksenine göre θ (saat tersi) = $\begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Yamultma (Shear) Dönüşümü

Düzlemde bir noktanın, bir koordinatını sabit bırakırken, diğer koordinatını sabit kalan koordinatın bir katı kadar değiştiren dönüşüme yamultma (shear) dönüşüme denir. Burada yapılan cismin bir koordinatının belirli bir yöne doğru belirli bir oranda arttırılmasıdır.

x ekseninde;

$$S_x: \mathbb{R}^2 \rightarrow \mathbb{R}^2, S_x(x, y) = (x + cy, y) \Rightarrow A = \begin{bmatrix} 1 & c \\ 0 & 1 \end{bmatrix}$$



y ekseninde c kuvvetinde yamultma;

$$S_y: \mathbb{R}^2 \rightarrow \mathbb{R}^2, S_y(x, y) = (x, y + cx) \Rightarrow A = \begin{bmatrix} 1 & 0 \\ c & 1 \end{bmatrix}$$

Büyültme ve Küçültme İşlemi

Bir vektörün doğrultusunu değiştirmeden, vektörün uzunluğunu küçüten veya büyüten dönüşümlere küçültme ve büyültme (contraction and dilation) dönüşümleri denir. Bir vektörün uzunluğunu c kat küçüten veya büyüten bir dönüşüm;

$$T: \mathbb{R}^2 \rightarrow \mathbb{R}^2, T(x, y) = (cx, cy) \Rightarrow A = \begin{bmatrix} c & 0 \\ 0 & c \end{bmatrix}$$

\mathbb{R}^2 de

$$\begin{bmatrix} c & 0 \\ 0 & c \end{bmatrix}$$

\mathbb{R}^3 'de

$$\begin{bmatrix} c & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & c \end{bmatrix}$$

NOT = Öteleme dönüşümü lineer değildir

Lineer Dönüşümlerin Bileşkesi

$T_A: \mathbb{R}^n \rightarrow \mathbb{R}^k$ ve $T_B: \mathbb{R}^k \rightarrow \mathbb{R}^m$ lineer dönüşümler o.ü

$$(T_B \circ T_A)(\vec{x}) = (T_B(T_A(\vec{x})))$$

dönüşümü de lineerdir ve bu ifade lineer dönüşümlerin bileşkesi denir.

Ör

$T_A: \mathbb{R}^3 \rightarrow \mathbb{R}^2$, $T_A(x, y, z) = (x+y, x-z)$ ve $T_B: \mathbb{R}^2 \rightarrow \mathbb{R}^3$,
 $T_B(x, y) = (2x, x-y, 3y)$ $T_A \circ T_B$ dönüşümünü bulun.

$$AB = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 1 & -1 \\ 0 & 3 \end{bmatrix} = \begin{bmatrix} 3 & -1 \\ 2 & -3 \end{bmatrix}$$

elde edilir ve $T_A \circ T_B(x, y) = (3x-y, 2x-3y)$ bulunur.

Ör

Verilen noktaya göre, verilen dönüşümlerin bileşkesi uygulandıktan sonra elde edilen yeni noktayı bulun.

i) $\vec{x} = (4, -1, -3)$ vektörünün 2 kat büyütülmesi ve y eksenine indirilmesi

cvp)

$$\begin{bmatrix} 4, -1, -3 \end{bmatrix}_{1 \times 3} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}_{3 \times 3} = \begin{bmatrix} 8, -2, -6 \end{bmatrix}_{1 \times 3}$$

$$\begin{bmatrix} 8, -2, -6 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0, -2, 0 \end{bmatrix}_{1 \times 3} \text{ olur.}$$

ii) $\vec{X} = (4, 2)$ vektörünün 45° döndürülmesi ve x eksenine izdüşümü

cvp

$$\begin{bmatrix} 4, 2 \end{bmatrix}_{1 \times 2} \begin{bmatrix} \cos 45 & -\sin 45 \\ \sin 45 & \cos 45 \end{bmatrix} = A$$

$$A \cdot \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 8 \\ 0 \end{bmatrix} \rightarrow \text{olur.}$$

3d bir resimde 45 derece döndürme y'ye göre;

```
>> x = [csurezd('logox.txt'); csurezd('logoy.txt');  
        csurezd('logoz.txt')];
```

```
>> plot3(x(1,:), x(2,:), x(3,:)), title('deneme3d');
```

```
>> theta = 45;
```

```
>> A = [cos(theta) 0 sin(theta); 0 1 0; -sin(theta) 0 cos(theta)];
```

```
>> R_y = A * x;
```

→ Bu yarı

axis tight

axis equal yap.

gibi yapılır

$$0th = zeroth$$

3 kez ölçölme için;

$$\gg A = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{bmatrix};$$

BÖLÜM 5: Numerical Techniques with MATLAB

Polynomizls

- Birçok fonksiyon, yüksek dereceli bir polinom ile iyi tanımlanabilir.
- Matlab'da polinomlar katsayılar (coefficient) vektörü olarak ifade edilir.
- Bir P vektörü polinomuz; $2x^3 + bx^2 + cx + d$
 $\uparrow \quad \uparrow \quad \uparrow \quad \uparrow$
 $P(1) \quad P(2) \quad P(3) \quad P(4)$
- $P = [1 \ 0 \ -2]$ ifadesi $x^2 - 2$ polinomunu gösterir.
- $P = [2 \ 0 \ 0 \ 0] \Rightarrow 2x^3$ olur \rightarrow 3. derece polinom.

Polynomial Operations

- P , $[N+1]$ unlagundz vektorse, N .th dereceden polinomun ifade eder,
- To get the roots of polinom

$\gg r = \text{roots}(P)$ r is a vector of length N

- Also we can get the polynomial from the roots

$\gg P = \text{poly}(r)$ r is a vector length N

roots bize bir polinomun köklerini veriyor.
poly ise köklere göre polinom buluyor.

$$x^2 - 4 = (x+2)(x-2)$$

$$\begin{bmatrix} 1 & 0 & -4 \end{bmatrix} \quad \begin{matrix} -2 \\ 2 \end{matrix}$$

$$>> P = [1 \ 0 \ -4]$$

$$>> r = \text{roots}(P)$$

$$r =$$

$$-2.000$$

$$>> x = \text{poly}(r)$$

$$x =$$

$$1.0000 \quad -0.0000 \quad -4.0000$$

- Belli bir noktadaki polinom değerini bulmak için;

$$>> y_0 = \text{polyval}(P, x_0)$$

x_0 is a single value, y_0 is a single value.

- Birçok noktayı hesaplamak için

$$>> y = \text{polyval}(P, x)$$

x bir vektör, y de aynı boyutta vektördür.

ÖR

$$x^2 - 2x = 0 \text{ için}$$

$$>> P = [1 \ -2 \ 0];$$

$$>> \text{polyval}(P, 2)$$

$$\text{ans} =$$

$$0$$

$$>> \text{polyval}(P, 1)$$

$$\text{ans} =$$

$$-1$$

$$>> \text{polyval}(P, [0, 1, 2, 3])$$

$$\text{ans} =$$

$$0 \quad -1 \quad 0 \quad 3 \text{ olur.}$$

$$2x^2 + 3x^2$$

- Sum of two polynomials:

```
>> f = [a0, a1, a2, a3];
>> g = [b0, b1, b2, b3];
>> r = f + g;
```

⚠ (sıraları eşit olmalı)

- multiply 2 polynomials

```
>> f = [1, 2, 3]; h = [2, 1, 2];
>> g = conv(f, h)
g =
    2    5   10    7    6
```

- polynomial division can be accomplished using deconv(f, h)

```
>> f = [2, 5, 10, 7, 6];
```

```
>> g = [1, 2, 2];
```

```
>> [h, r] = deconv(f, g)
```

```
h = 2 1 4
```

```
r = 0 0 0 -3 -2
```

→ buna tek de zayıfladık

Interpolation and Curve Fitting (Eğri uydurma)

Verilen veri noktalarında $\{x_i, y_i\}_{i=1}^n$ ile ilgilenilen tüm noktaların miktartanımıyla bir $f(x)$ fonksiyonu olsun. Burada fonksiyonu uydurmaya çalışacağız.

Interpolation

- Find a function satisfying

$$P(x_i) = f(x_i), \quad i = 1 \dots n$$

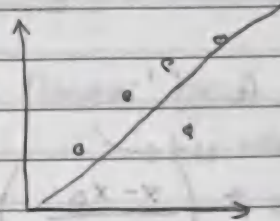
- that allows us to approximate $f(x)$ such that the function values between the data sets may be estimated.

- Interpolasyonda veri noktalarından geçen bir fonksiyon bulmaya çalışılır. (tam üzerinden)

Curve Fitting

- Find a function that is a good fit to the original data points. The function does not have to pass through the original data points.

- Eğri uydurmadız, uydurulan eğride, verilen noktaların üzerinden geçme zorunluluğu yoktur.



gibi olabilir.

- ★ With interpolation we search a function that allows us to approximate such that functional values between the original data set values may be estimated.

- The interpolation function typically passes through the original data set.

- With curve fitting we simply want a function that is a good fit to the original data points.

- With curve fitting the approximating function does not have to pass through the original data set.

1) Linear Interpolation

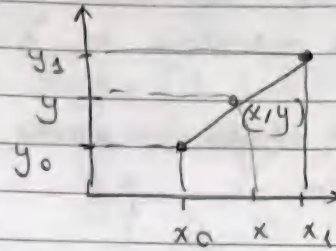
- The simplest type of interpolation is linear interpolation, which simply connects each data point with a straight line.

- The polynomial that links the data points together is of first degree, eg, a straight line.

- Given two points (x_0, y_0) and (x_1, y_1) , find $P(x)$ such that

$$P_0 = f(x_0)$$

$$P_1 = f(x_1)$$



P_1 can be written in the form:

$$y = \left(1 - \frac{x - x_0}{x_1 - x_0}\right) y_0 + \left(\frac{x - x_0}{x_1 - x_0}\right) y_1$$

MATLAB'de

$yq = \text{interp1}(x, y, xq, \text{method})$

- $x \rightarrow$ sample values) *gerçek değerler*
- $y \rightarrow$ values $f(x)$
- $xq \rightarrow$ query points on which the polynomial will be evaluated.
- method \rightarrow method of interpolation (linear, cubic vb)

ör

Time (s)	Velocity (km/h)
0	0
1	10
2	20
3	30
4	40
5	50

$x = 0:5;$

$y = [0 \ 10 \ 20 \ 30 \ 40 \ 50];$

$xq = 0:0.1:5; \% 0'dan 5'e$

$yq = \text{interp1}(x, y, xq, 'linear');$

$\text{plot}(x, y, 'o', xq, yq);$

x ve y
noktalarını
marker
olarak çiz

Bu 0.01'ler.

tem bulunuşları
hem sampleleri
çizel.

Hızlı ihtimali var.

ÖR

Sinüs eğrisini interpolasyonla yapmayı bakalım

$x = 0:10;$

$y = \sin(x);$

$xq = 0:0.25:10;$

$yq = \text{interp1}(x, y, xq, 'linear');$

$\text{plot}(x, y, 'o', xq, yq);$ \rightarrow her nokta ayrı doğru olur. (A

sinüste sarmalar çok olur.

Daha hassas interpolasyon işlemi yapabilir miyiz?

2) Cubic spline Interpolation

- Connecting data points with straight lines probably isn't the best way to estimate intermediate values

- An improved interpolation technique is to replace the straight line connecting the data points with a third-degree polynomial.

- The third degree polynomial is of the form $y = f(x) = \alpha_3 x^3 + \alpha_2 x^2 + \alpha_1 x + \beta$

$$y = f(x) = \alpha_3 x^3 + \alpha_2 x^2 + \alpha_1 x + \beta$$

Aynı şeyi cubic spline ile yaparsak daha iyi sonuç alırız.

$yq = \text{interp1}(x, y, xq, 'spline');$

$\text{plot}(x, y, 'o', xq, yq), \text{grid on};$

probleme göre seçilir.

Interpolasyon Gesitleri

'linear' \rightarrow default -

'nearest' \rightarrow nearest neighbor interpolation

'spline' \rightarrow piecewise cubic spline interpolation

'cubic' \rightarrow cubic interpolation

'pchip' \rightarrow shape preserving piecewise cubic interpolation

1) Curve Fitting

- We could use interpolation techniques to find values of y between measured x -values,
- However, it would be more convenient if we could model experimental data as $y=f(x)$
- MATLAB has built-in curve-fitting functions that allow us to model data empirically.

★ Doğru arzılıktaki data almak önemli.

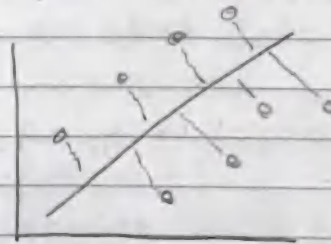
e) Least-Squares Curve Fitting

(En küçük kareler)

- A least squares curve fit can be used to obtain a curve such that the squared distance from each point to the curve is minimized.

Curve can be

- polynomial of degree n
- trigonometric
- exponential



herbir noktaya arasındaki karesel hataları minimize edecek eğri aranır.

b) Linear Regression

• With linear regression a linear equation, $y = mx + b$, is chosen that fits the data points such that the sum of the squared error between the data points and the line is minimized.

• The squared distance is computed with respect to the y axis.

• Given a set of data points (x_k, y_k) , $k = 1 \dots N$, the mean squared error (MSE) is defined as:

(ortalama karesel hata)

$$MSE = \frac{1}{N} \sum_{k=1}^N (y_k - f(x_k))^2$$

$\nearrow (mx_k + b)$

Uygurduğumuz eğrinin verdiği değer ile gerçekte okunmuş olduğumuz verinin farklarının karelerinin toplamını minimize etmeye çalışıyoruz.

y = okunan data $f(x)$ = eğrinin o noktada verdiği değerdir
— arasındaki fark minimum olsun istiyoruz.

c) Polynomial Regression

$$f(x) = a_1 \cdot x^n + a_2 \cdot x^{n-1} + \dots + a_{n-1}$$

fits the data

• MATLAB'de, polyfit fonksiyonu ile sağlanır.

$$a = \text{polyfit}(x, y, n)$$

a = row vector of coefficients of the regression

x ve y = sağlanan veri noktaları

n = polinom derecesi

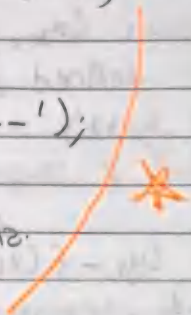
ör

$X = [-1 \ 0 \ 2]$ ve $Y = [0 \ -1 \ 3]$ olsun

finds the best second order polynomial that fits the points $(-1, 0)$, $(0, -1)$ ve $(2, 3)$

```
>> X = [-1 0 2]; Y = [0 -1 3];  
>> p2 = polyfit(X, Y, 2); → Polinom buluyor  
>> plot(X, Y, 'o', 'MarkerSize', 10);  
>> hold on;  
>> x = -3:0.01:3;  
>> plot(x, polyval(p2, x), 'r--');
```

↑
p2 polinomun
fonksiyon
x değeri için hesapla.



ör

$x = -4:0.1:4$ 'n reszplz

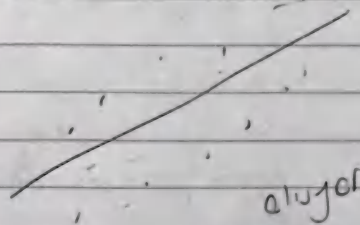
• Add random noise to these samples. (randn kullanarak)

Noisy signal 'i • markerlyz plotla (2. derece) yudur

• Plot the fitted polynomial on the same plot, using the same x values and a red line.

ör

```
>> x = -4:0.1:4;  
>> [m, n] = size(x);  
>> y = x + randn(m, n);  
>> p = polyfit(x, y, 2);  
>> plot(x, y, 'o');  
>> hold on  
>> plot(x, polyval(p, x), 'r');
```



OR

```
>> x = 0:5;
```

```
>> y = [0 10 25 36 52 59];
```

```
>> x_ls = 0:0.01:5;
```

Linear fitting

```
>> figure; subplot(2 2 1); plot(x, y, 's'); hold on;
```

→ bitişik de olabilir.

→ square (kare)

```
>> p1 = polyfit(x, y, 1);
```

```
>> y1 = polyval(p1, x_ls);
```

```
>> plot(x_ls, y1);
```

```
>> axis([0 5 0 60]); grid on;
```

```
>> title('First Degree');
```

second degree fitting

```
>> p2 = polyfit(x, y, 2);
```

```
>> y2 = polyval(p2, x_ls);
```

```
>> subplot(2 2 2); plot(x, y, 's'); hold on;
```

```
>> plot(x_ls, y2);
```

```
>> title('Second Degree');
```

fifth degree

```
>> p5 = polyfit(x, y, 5);
```

```
>> y5 = polyval(p5, x_ls);
```

```
>> subplot(2 2 3); plot(x, y, 's'); hold on;
```

```
>> plot(x_ls, y5);
```

```
>> title('Fifth Degree');
```

5. derece daha smooth daha iyi oldu.

10. derecede ise subplot (2 2 4) için > mantıksız oldu. Her zaman yüksek dereceden eğri iyi olmaz.

yani datayı tanımak gerek. En fit dereceli eğriyi bulmalıyız.

Interpolation in 2D

$vq = \text{interp2}(x, y, V, xq, yq, \text{method})$
↓
2 boyutlu

$(x, y) \rightarrow$ 2 boyutlu V 'de bunlara denk gelen değer.

V 'yi bilmiyoruz.

Farklı tür metotlar kullanılabilir.

• cubic, spline, nearest gibi.

• $z = \text{peaks}(x, y)$ evaluates the peaks function at x and y .

• We can use z as data points and visualize interpolation using various methods.

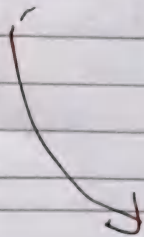
ör

```
[x, y] = meshgrid(-4:4);
```

```
z = peaks(x, y);
```

```
surf(x, y, z);
```

```
title('Original Data');
```



6 Derzmi

% % Linear Interpolation

```
>> [xq, yq] = meshgrid(-4:0.25:4);
```

```
>> Vq = interp2(x, y, V, xq, yq, 'linear');
```

```
>> surf(xq, yq, Vq); // Daha yumuşak görümler oldu.
```

```
>> title('Linear interpolation');
```

cubic yapınca daha kırılmı,
spline daha iyi olur.

Image Manipulation

- Starting with a random image $im = \text{rand}(10, 10)$;

- Interpolate the image using 64 times as many points in each direction.

QR

% % Generate Random Image

```
>> im = rand(10, 10);
```

```
>> figure; subplot(2, 2, 1);
```

```
>> imshow(im);
```

```
>> title('original');
```

% % Interpolation

```
>> [m, n] = size(im);
```

```
>> [x, y] = meshgrid(1:m);
```

```
>> [xq, yq] = meshgrid(linspace(1, m, 64 * m));
```

```
>> im_linear = interp2(x, y, im, xq, yq, 'linear');
```

```
>> im_cubic = interp2(x, y, im, xq, yq, 'cubic');
```

```
>> im_spline = interp2(x, y, im, xq, yq, 'spline');
```

% % Show Images

```
>> subplot(2, 2, 2);
```

```
>> imshow(im_linear);
```

```
>> title('linear interpolant');
```

```
>> subplot(2, 2, 3);
```

```
>> imshow(im_cubic);
```

```
>> title('Cubic');
```

```
>> subplot(2, 2, 4);
```

```
>> imshow(im_spline);
```

```
>> title('Spline');
```

Öl (Sinev Sorusu)

Write a function named checkerboards(nrows, ncols, n) that draws a $n \times n$ checkerboard according to the user-specified dimensions. Your function must accept three arguments, nrows specifies the number of rows of the checkerboard, ncols specifies the number of columns and n specifies the number of blocks for each row and column. Your function must display a checkerboard by using imagesc function and grayscale colormap. Your function must display a checkerboard by using imagesc function and a grayscale colormap. Your function must return a 2D matrix that represents the checkerboard.

Mesela (4, 4, 4) için

4, 4, 2 için

1 0 1 0

0 1 0 1

1 0 1 0

0 1 0 1

1 1 0 0

1 1 0 0

0 0 1 1

0 0 1 1

olmaz

Benim cevabım

```
function [board] = checkerBoard3(nrows, ncols, n)
```

```
block_rows = floor(nrows / n);
```

```
block_cols = floor(ncols / n);
```

```
block = ones(block_rows, block_cols);
```

```
nanblock = zeros(block_rows, block_cols);
```

```
mat = [block; nanblock]';
```

```
flipmat = flip(mat, 2);
```

```
newmat = [mat; flipmat];
```

```
board = repmat(newmat, nrows / length(newmat), ncols / length(newmat));
```

```
imagesc(board);
```

```
colormap('gray');
```

```
end
```

BÖLÜM-6 IMAGE PROCESSING WITH MATLAB

Resim nedir?

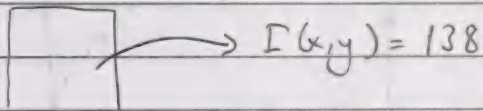
- In its most general form, an image is a function f from $\mathbb{R}^2 \rightarrow \mathbb{R}$

- $f(x,y)$ give the intensity of 2 channel at position (x,y)

```
>> I=imread(256, 256);
```

```
>> imshow(I);
```

- Each part of an image is a pixel.



- Signal = fonksiyon bazı fiziksel anizimli değişkenlere bağlıdır.

- Image = Bu fonksiyon ile örnekleme

- 2 variables: xy coordinates

- 3 variables: xy + time (video)

- 'Brightness' is the value of the function for visible light.

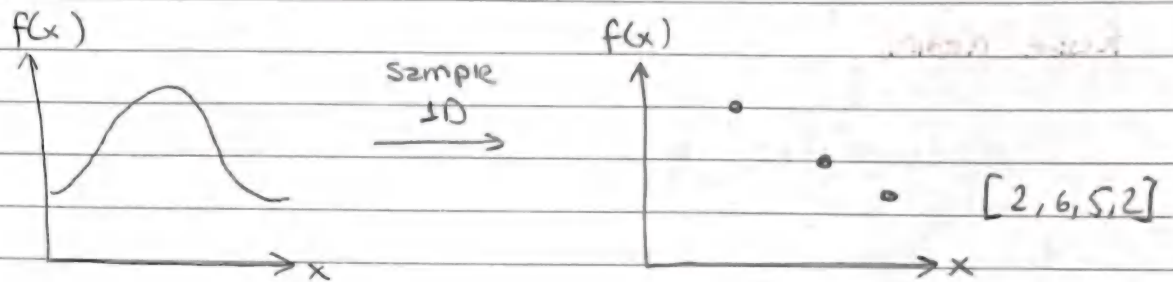
- Can be other physical values too: temperature, pressure, depth.

Görüntü işleme dediğimiz digital bir veri haline getirilmesi esidir.

(ultrasonik, termal, mr ---) (Görme, X-Ray, UV, visible light, IR, Radio waves)
(ışık spektrumu)

Sampling In 1D

- Sampling in 1D takes a 'function', and returns a vector whose elements are values of that function at the sample points



Sampling in 2D

- Sampling in 2D takes a function and returns a matrix

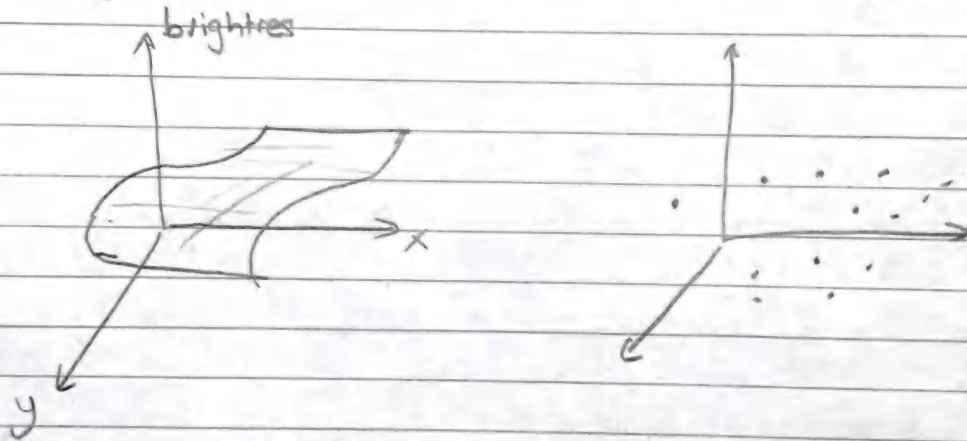
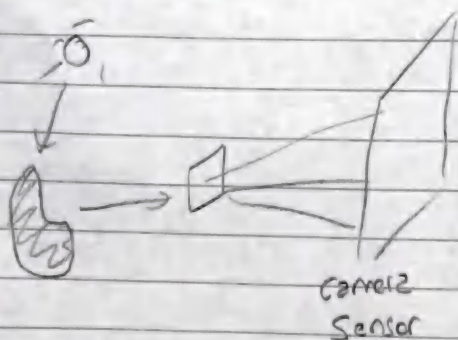
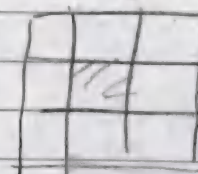


Image Formation



- Integrating light over a range of angles

Output Image



gibi olur.
Buzlar
pikseller
(yoğunluk ölçülerek
yapılır.)

Aktif sensörler = Çalıştıkları yüzeye sinyal göndererek
bunu algılayarak. (LIDAR meselesi)

Resolution

Geometric vs Spectral Resolution

→ En küçük mesafe birimi genelde piksel için kullanılır.

500x500 iki resim olsun

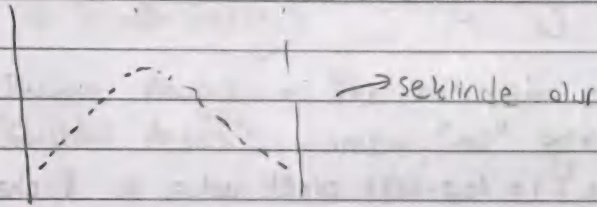
İki resim arasındaki boyut ve piksel görünümlüğü önemlidir.

Geometrik → uydudan sensörle.

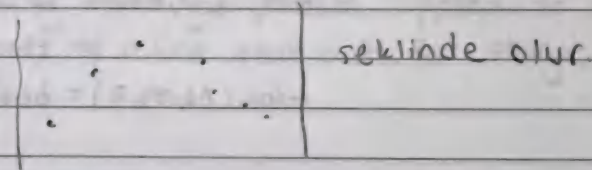
Quantization

Bir resmin

Brightness'ı önce 27 sonra 255 sonra 256 tekler 255.



Quantization yaptığımızda



Radiometric Resolution

Quantization effectidir. Her bir piksellik verinin kaç bitlik veriye gösterileceğini ifade eder.

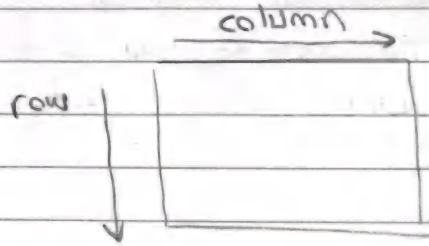
8 bit → 256 , 4 bit → 16 levels , 2 bit → 4 levels
1 bit → 2 levels

8 → 1 bite doğru değeri azalır.

Images In Matlab

• $N \times M$ image "im"

- $im(1,1)$ = top-left pixel value
- $im(y,x)$ = y pixels down, x pixels to right
- $im(N,m)$ = bottom-right pixel



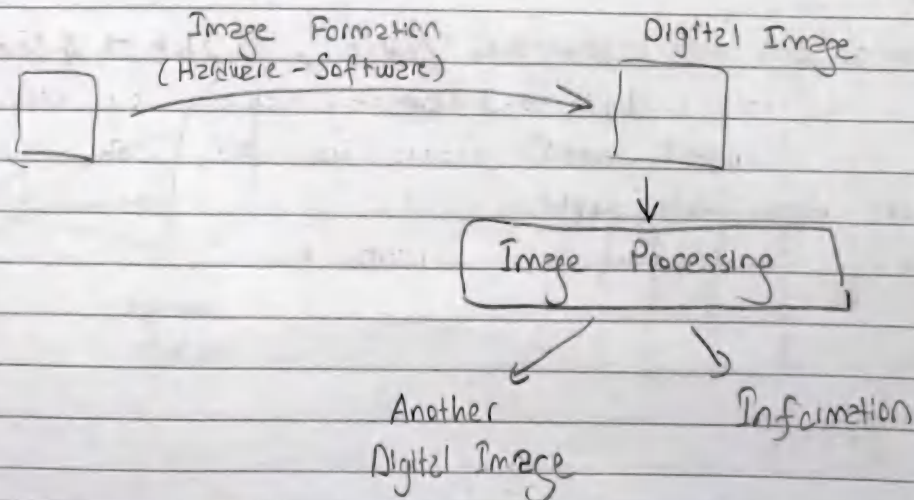
Grayscale Intensity, RGB (3 tone Grayscale Gbri)

• $N \times M$ RGB image "im"

- $im(1,1,1)$ = top-left pixel value in R-channel.
- $im(y,x,b)$ = y pixels down, x pixels to right in the bth channel.
- $im(N,m,3)$ = bottom-right pixel in the B channel.

Image Processing

- Image processing: operations that take images as input, produce images and information as output.



corrupt=noisy

→ mesela uzaktaki okuyucuları

Image Processing, ^{computer photography} → signal processing, computer vision, graphics, machine learning, statistics, applied math gibi alanlarda kullanılabılır

• The processing of digital images can be divided into several classes:

- Image enhancement = the process of improving the quality and the (görüntü iyileştirme) information content of images so that the results are more suitable for display or further image analysis.

- Image restoration = the operation of taking a corrupt/noisy image (resimdeki gürültüler, vb ortadan kaldırılır) and estimating the clean, original image.

- Image Analysis = The extraction of meaningful information from (Görüntü Analizi) images

- Image compression = type of data compression applied to digital (Görüntü sıkıştırma) images, to reduce their cost for storage or transmission

Computer Vision (Bilgisayarlı Görü) = Amaç, makineye, insan gözüymüş gibi bir algılama kapasitesi kazandırmak. (ör. insan tespiti)

Images In Matlab

• Reading images: imread open and read the contents of image files in most popular formats (eg. TIFF, JPEG, BMP, GIF and PNG etc)

```
>> im = imread('faculty.bmp');
```

• Display an image

```
>> figure; imshow(im);
```


- Writing images: `imwrite` writes image data to the file specified by filename. (kaydetme)

```
>> imwrite(im, 'faculty.bmp');
```

Image Viewer Application

- `imtool`: displays an image and contains several associated tools that can be used to explore the image contents.

```
>> imtool(im) ← Resim pikselleri hakkında bilgi verir.
```

Resmin bulunduğu dosya açılır

MATLAB'de

```
>> im = imread('faculty.jpg');
```

$im = 485 \times 780 \times 3$ olur → RGB channeleri belirtir. (uint8 olarak alıyor.)

(x,y) boyutlarında piksel bulunur. 2 boyuttan tek boyuta çekilebilir

Tek RGB yok.

time → tek bir görüntü zamanla değişen görüntüler için kullanılır

```
>> imR = im(:, :, 1); // Sadece kırmızı kanalı alır.
```

```
>> figure; imshow(imR); // Siyah beyaz ama bu kırmızıyı ifade eder
```

2 için → 6, 3 için 8 çıkar

→ yüklemesi uzun sürüyor

```
>> imtool(im); // Resimde gezinirken piksel değerlerini gösteriyor.
```

```
// Inspect Pixel values'e tıklayarak daha detaylı görünür.
```

```
>> imwrite(im, 'faculty.bmp');
```

Resmi sızdırma, veya yukarıdaki resmi çevirmek için

```
>> imlr = flip_lr(im);  
>> imud = flipud(im);  
>> figure; imshow(imud);  
>> figure; imshow(imlr);
```

Dat2 Classes and Dat2 Conversions

- To convert an image to a dat2 class and range suitable for image processing, you can use one of the functions listed in the table

Name	Dat2 Class	• The most common dat2 classes for images are as follows:
im2single	single	
im2double	double	
im2uint8	uint8	- uint8: 1 byte per pixel, in the [0, 255] range
im2uint16	uint16	- double: 8 bytes per pixel, [0.0, 1.0] range
im2int16	int16	- logical: 1 bit per pixel, 1 → white 0 → black

Bir RGB resmi grayscale'e çevirmek için

- rgb2gray → convert an RGB image to grayscale ($\frac{R+G+B}{3}$ her pixel için yapılabilir)

```
>> img = rgb2gray(im);  
>> figure; imshow(img);
```

Pratikte böyle olmuyor.

Bir ağırlıklı ortalamayla yapılıyor.

$$\left(\begin{array}{l} \text{rgb2gray} \\ 0.2989 \cdot R + 0.5870 \cdot G + 0.1140 \cdot B \end{array} \right)$$

Görüntü pikellerini double'a çevelim

>> imD = im2double(imG); // 0 ile 1 arasında çevilmiş olur.

0 ile 255 arasında çevilsin ama double olsun istiyorsak

>> imDD = double(imG); // imDD = imDD + 0.1; yapılabilir.

Image Resizing

- `imresize(image, scale)`: returns an image which is scale times of the given image.

>> imR1 = imresize(im, 0.5, 'nearest');

>> imR2 = imresize(im, 0.5, 'bilinear');

>> imR3 = imresize(im, 0.5, 'bicubic');

>> figure, subplot(1,3,1), imshow(imR1), title('Nearest-Neighbor Interpolation');

>> subplot(1,3,2), imshow(imR2), title('Bilinear Interpolation');

>> subplot(1,3,3), imshow(imR3), title('Bicubic Interpolation');

// bicubic 2 kat daha pürüzsüz bilinear göre daha okunaklı oluyor
// illeki bir kopya olur.

Image Rotation

- `imrotate(image, angle)` = rotates image by angle degrees in a counterclockwise direction.

- When you rotate an image, you specify the image to be rotated and the rotation angle, in degrees

- Like `imresize`, `imrotate` allows the user to specify the interpolation method used: nearest-neighbor (default), bilinear, or cubic

```
>> imRot = imrotate(im, 45, 'bilinear');
```

Döndürme demek aslında orijinal piksel değerlerin ortaya çıkması demek (sizin tersi)

Image Cropping

- `imcrop(image)`: displays the image in a figure window and creates an Interactive Crop Image tool associated with an image.
- `imcrop(image, rect)` → crops the image according to the position and dimensions specified in the crop rectangle (`[x, y, width, height]`)

```
>> imCropped = imcrop(im);
```

```
>> imCropped = imcrop(im, [40, 40, 200, 200]);
```

↓

bunu şöyle de yapabiliriz

```
>> imCropped = im(40:240, 40:240, :);
```


BÖLÜM 7 IMAGE FILTERING DEVAM

Image Filtering

Compute function of local neighborhood at each position.

$$h(m,n) = \sum_{k,l} f(k,l) I(m+k,n+l)$$

• Amacı

- Enhance images = Denoise, resize, increase contrast, etc.
- Extract information from images = texture, edges, distinctive points, etc.
- Detect patterns: template matching
- Deep Convolutional Networks

Averaj filtre = $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

Filtreleme (konvolüsyon), pencerenin görüntü üzerinde gezdirilmesi işlemidir. (Görüntü işleme bkz.)

Bir görüntüye uyguladığımızda düşük frekanslı geçer, yüksek frekanslı bastırır.

⊗ → konvolüsyon işareti.

→ filtre

$$f[x,y] \otimes g[u,v] = h[x,y]$$

görüntü kenarları kaybedilir. Bunun için çeşitli yöntemler var.

Box Filter

- Pencerenin dahilindeki piksellerin ortalamasını alıp merkez pikselin yeni değerini değiştirir. Bu smoothing effecttir. (yumuşatma yani keskin kenarları azaltır.)

Bütün değerler aynı olur

Gaussian Filter

→ Yakın komşu daha etkili.

1	1	2	1
16	2	4	2
	1	2	1

σ (sigma) arttıkça bulanıklık artar.
Gaussian Function
$$g(u,v,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$

yzpı korunur. ama texture (detaylar) kaybedilir.

Smoothing with Gaussian Filter

- The amount of smoothing ^(bulanıklaştırma) depends on the value of the spread parameter (σ)

for sigma = 1:3:10

```
h = fspecial('gaussian', fsize, sigma);  
out = imfilter(im, h);  
imshow(out);  
pause;
```

end

İnsan gözünün bulanıklığı algılaması açısından gaussian daha iyidir

acentuztes = vurgular

Practice with Linear Filters

0	0	0
0	1	0
0	0	0

→ resmin aynısını verir.

0	0	0
0	0	1
0	0	0

→ sola kayar. (Shifted left By 1 pixel)

0	0	0
0	2	0
0	0	0

$-\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$-\frac{1}{9} \quad -\frac{1}{9} \quad -\frac{1}{9}$
 $-\frac{1}{9} \quad 2 \quad -\frac{1}{9}$
 $-\frac{1}{9} \quad -\frac{1}{9} \quad -\frac{1}{9}$

Keskinleştirir.

Sharpening Filter

local average ile olan farklılıklar vurgulanır.

Yüksek frekansları çıkarır.

Sobel Filter

Yüksek geçiren filtredir.

1	0	-1
2	0	-2
1	0	-1

Kenar tespit operatörüdür.

Yataydaki değişimleri bulduğu için bir dikey
kenar tespit operatörüdür.

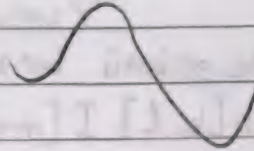
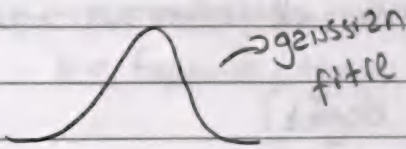
Yatay tespit için transpozunu alınız

1	2	1
0	0	0
-1	-2	-1

prewitt'de

1	1	1
0	0	0
-1	-1	-1

2D Edge Detection Filters



Gaussian'ın
türevi $\frac{\partial}{\partial x} h(x, y)$

Laplacian ∇^2 operatör (sum of 2nd derivatives) (filtrenin türevi)

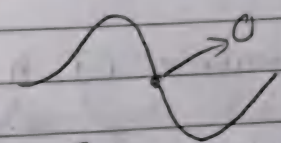
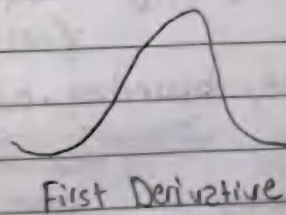
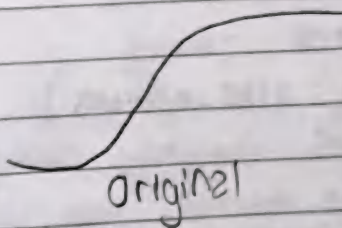
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Genelde

0	1	0
1	-4	1
0	1	0

→ Toplamı 0

Kenarları buldurun



Kenar tespit edilir

1	1	1
1	-8	1
1	1	1

→ Bu da Laplacian filterdir

Correlation and Convolution

Uyguladığımız filtre simetrik ise yindirler

Correlationda 90° döndürülür. Konvolüsyonda döndürülmez.

2D Correlation (Amacı filtre ile görüntü benzerliğini ölçmek)

$$Y[m,n] = \sum_{k,l} h[k,l] I[m+k,n+l]$$

$$Y = \text{filter2}(h, I); \text{ or } Y = \text{imfilter}(I, h);$$

2D Convolution (Filtreyi uygulayarak görüntüde değişimler yapmak)

$$Y[m,n] = \sum_{k,l} h[k,l] I[m-k,n-l]$$

$$Y = \text{conv2}(h, I); \text{ or } h = \text{imfilter}(I, h, 'conv');$$

$$\text{conv2}(I, h) = \text{filter2}(\text{rot90}(h, 2), I)$$

imfilter

• Linear filters can be implemented in MATLAB by using imfilter function.

$$Y = \text{imfilter}(I, h, \text{mode}, \text{boundary_options}, \text{size_options});$$

input image

conv
korelasyon
filter mask

(büyüklüklerine
işin kenar
pikseller kaybedilebiliyor)

derste
konvolüsyon
yapılacak

Bunun için

'symmetric' (simetrisini ekler)

'replicate' (ayrısını koyar)

'circular'

yapılabilir

size options=

- 'full' → original görüntü padding edilmiş / extend edilmiş boyutta olur.
- 'same' → output görüntü, input görüntü boyutunda olur.

fspecial

- fspecial: simply create predefined 2D filters

`h = fspecial(type, parameters)`

type sunlar olabilir:

- 'average' → Averaging filter
- 'disk' → circular averaging filter
- 'gaussian' → gaussian low-pass filter
- 'laplacian' → 2D laplacian operator
- 'log' → laplacian of Gaussian (LoG) filter
- 'motion' → Approximates the linear motion of camera
- 'prewitt' and 'sobel': horizontal edge-emphasizing filters
- 'unsharp': unsharp contrast enhancement filter.

Örnekler

ImageFiltering.m 2dinde script açılır

```
% % Read Image
```

```
im = imread('panda.jpg');
```

```
figure; imshow(im);
```

//genelde grayscale üzerinden filtreleme yapılır. Öbür türlü R,G,B için
// ayrı ayrı filtreleme yapılmalı.

```
im = rgb2gray(im);
```

```
im = im2double(im);
```


% % low-pass filters

h_box = fspecial('average', [13 13]); // genelde tek sayı verilir ki
// ortadaki sayı alınabilirsin.

// kendimiz üretmek istersek

% h_box = ones(13) ./ (13 * 13);

im_box = imfilter(im, h_box, 'conv');
figure; imshow(im_box);

h_gauss = fspecial('gaussian', [13 13], 0.5);

im_gauss = imfilter(im, h_gauss, 'conv');

figure; imshow(im_gauss);

→ sigma arttıkça bulanıklık azalır.

% % high-pass filters

h_lap = [0 -1 0; -1 4 -1; 0 -1 0]; // Laplacian filter

im_lap = imfilter(im, h_lap, 'conv');

figure; imshow(im_lap);

-1 8 -1 daha iyi ama güçlüdür elyaz

* Command Window'a

>> sum(h_gauss(:)) → 1 çıkar

h_box → 1

h_lap → 0 çıkar

Laplacian kullanılarak görüntüyü keskinleştirme:

im_sharp = im + im_lap;

figure; imshow(im_sharp);

% % Edge Detection

h-sobel = fspecial('sobel'); → transpose

im-sobelH = imfilter(im, h-sobel, 'conv');

im-sobelV = imfilter(im, h-sobel', 'conv');

figure; imshow(im-sobelH);

figure; imshow(im-sobelV);

im-edge = im-sobelH + im-sobelV;

figure; imshow(im-edge); // hem yatay hem dikey kenar tespiti.

$$\text{image gradient magnitude} = \sqrt{g_y^2 + g_x^2} \quad \text{Yönü} = \theta = \tan^{-1} \left[\frac{g_y}{g_x} \right]$$

→ yön zıttır

[magnitude, orientation] = imgradientz(im-sobelH, im-sobelV);

figure; imshow(magnitude);

figure; imshow(orientation, [I]); → ölçekleme yapar

% % Other Filters

h-motion = fspecial('motion', 20, 45);

im-motionV = imfilter(im, h-motion, 'conv');

figure; imshow(im-motion);

length (bu hareketin büyüklüğünü belirtir.)
angle (sırt yönünün tersi)

90 için
yukarıdan aşağıya
hareket olur.

0 derece sağdan sola

BÖLÜM 7 BINARY IMAGES

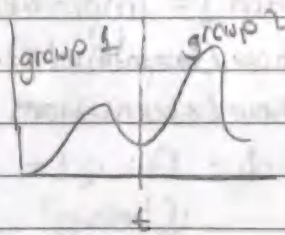
- In a binary image, each pixel assumes one of only two discrete values: 1 or 0.

- A binary image is stored as a logical array in MATLAB

Thresholding

- Binary images are often produced by thresholding a grayscale or color image, in order to separate an object in the image from the background.

- Thresholding provides an easy and convenient way to separate out the regions of the image corresponding to objects in which we are interested, from the regions of the image that correspond to background



- In simple implementation, each pixel in the image is compared with a threshold. If the pixel's intensity is higher than the threshold, the pixel is set to white in the output. If it is less than the threshold, it is set to black.

Threshold tespiti için belirli yöntemler mevcut.

Bunlardan biri Otsu's Methodu.

contribute = katkıda bulunur.

Automatic Thresholding: Otsu's Method

126bookpages'den okumayı önerir.

- Assumption: the histogram is bimodal
- Method: find the threshold t that minimizes the weighted sum of within-group variances for two groups that result from separating the grey tones at value t .

>> $t = \text{graythresh}(im)$ % image threshold using Otsu's method

Mathematical Morphology

- The field of mathematical morphology contributes a wide range of operators to image processing all based around a few simple mathematical concepts from set theory.
- The operators are particularly useful for the analysis of binary images and common usages include edge detection, noise removal, image enhancement and image segmentation.
- Most common morphological operators are:
 - dilation (genişletme)
 - erosion (silme)
 - closing (kapama)
 - opening (silme)

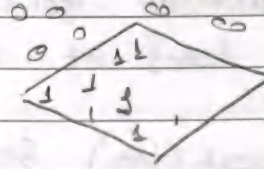
Siyah beyaz resimler üzerinde bu işlemler yapılarak görüntü daha iyi hale getirilebilir.

Bu işlem için belirli structuring element'e ihtiyacımız vardır.

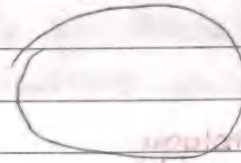
Structuring Elements

(filtre gibi) elimizde bir yapısal elemanımız var ve matematiksel operasyonları bunu kullanarak gerçekleştirebiliriz. Farklı tipte olabilir. MATLAB'de strel (structuring element) fonksiyonu ile her bir bu yapısal maskeler üretilebilir.

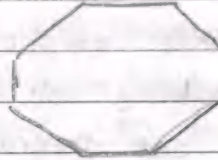
$SE = \text{strel}('diamond', r)$



$SE = \text{strel}('disk', r, n)$



$SE = \text{strel}('octagon', r)$

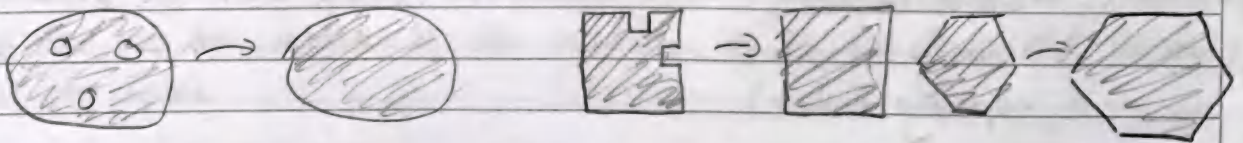


Ayrıca artırılabilir.

r, n şeklin boyutlarını belirler. (Şekillerin içi 1 dışı 0 olur.)

1) Dilation

Dilation birbirine bağlı olan 1'leri genişletmek için kullanılır. Delikler kapatılabilir. Özellikleri büyütmek için kullanılabilir.



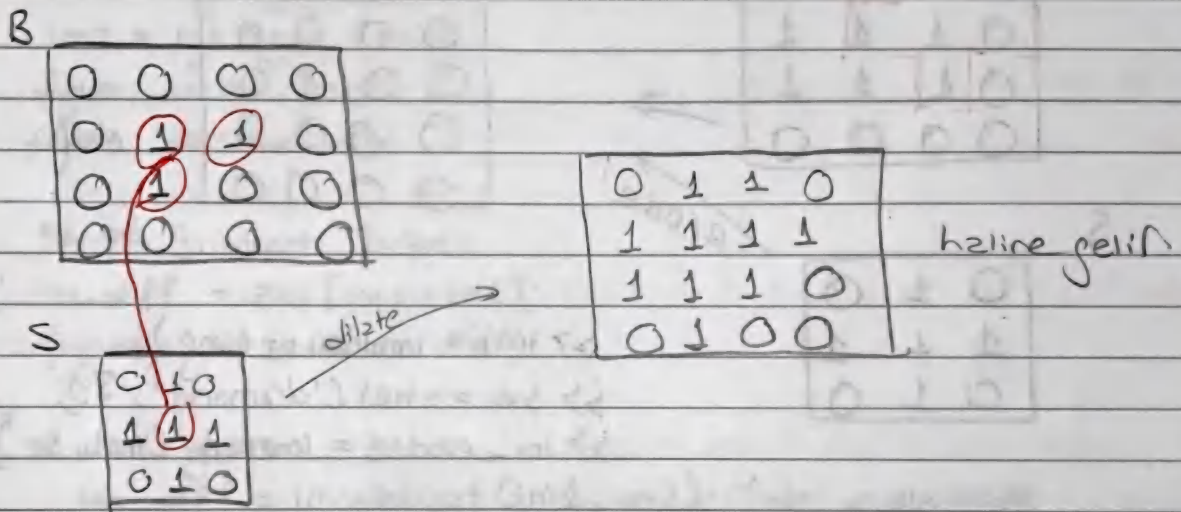
Dilation with Structuring Elements

Bunlar 12m tersidir.

The arguments to dilation and erosion are

- a binary image B
- a structuring element S

Dilate fonksiyonunda S, B'de getirildiği zaman S'in merkezindeki 1, B'deki 1'lerden biriyle karşılaştığı zaman OR işlemine tabi tutulur.



Dilation ile oyuklar giderilebilir.

>> imb = imbinarize(img); % siyah beyaz

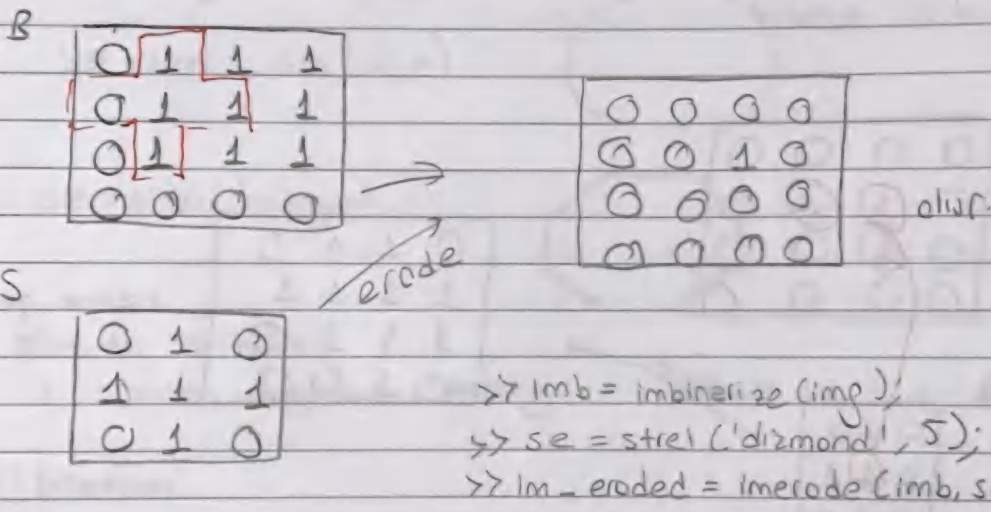
>> se = strel('diamond', 5);

>> im_dilated = imdilate(imb, se);

2) Erosion

Erosion ile küçültme (shrink) yapılır. Çukurluklar giderilebilir.
Featuveler, kırılmalar, bozulanlar kırmak için kullanılabiliriz.

S structural elementi, binary resim üzerinde gezdirilirken S'in tüm 1'leri, binary resimde de 1 ile karıştırmaz bir şey yapılmaz. Yoksa hepsini 0'lar.



3) Opening and Closing

Closing, \rightarrow önce dilation, sonra erosion

opening \rightarrow önce erosion, sonra dilation

Connected Components Labeling

Birbirini takip eden opening ve closing işlemleriyle beraber kenardaki algılardan kurtulur sadece içler kalır.

Örnekler

script

% % Read in Images

im1 = imread('Test1.jpg');

im2 = imread('Test2.jpg');

figure; imshow(im1);

figure; imshow(im2);

% % Convert to Grayscale

im1 = rgb2gray(im1);

im2 = rgb2gray(im2);

figure; imshow(im1);

figure; imshow(im2);

% % Subtract Images

// im_diff = abs(im1 - im2);

figure; imshow(im_diff);

Bunun yerine

im_diff = imsubtract(im1, im2); 'de yapılabilir.

Aynı.

% %

imhist(im_diff) ile histogram değerine bakılabilir.

Histogram = görüntüdeki her bir piksel değerine sahip kaç tane piksel var? anı gösterir.

% % Thresholding

im_diff = im2double(im_diff);

th = graythresh(im_diff); % Otsu yöntemi (elle de girilebilir)

im_bw = im_diff > th; % ile sayıya beyaz görüntü olur.

figure; imshow(im_bw);

Bunun yerine im_bw = imbinarize(im_diff); yazılabilir.

% % Fill in Regions

```
se = strel('disk', 3); % Gapi 3 7 ye 7 'ige kısıtlı gelir.  
im-open = imopen(im-bw, se); % derletme yapar. im close yapılabilir  
figure; imshow(im-open);
```

% Gırsıltılerden kurtulmek için; morfolojik operatör kullanılmış olur.
'disk', 1
imopen yapılır.

```
im_2rez = bw2rezopen(im-bw, 15); // 15pixel'den daha küçük  
olan bölgeler gider.
```

% % Region Properties

*

```
im-stats = regionprops(im-open, 'Major Axis Length');
```

*

```
im-length = [im-stats.Major Axis Length];
```

80 üzeri dersem bu bir nesnedir denilebilir.
(4 bölge var.)

```
idx = im-length > 80;
```

```
im-stats-final = im-stats(idx);
```

```
disp(im-stats-final);
```

% % Determine if change is significant

```
if isempty(im-stats-final)
```

```
disp('Nothing here');
```

```
else
```

```
disp('Something here');
```

```
end
```

Vize E-4 cevabı

$x = -15:0.1:15;$

$y = -15:0.1:15;$

$[X,Y] = \text{meshgrid}(x,y);$

$Z = \sin(\sqrt{X.^2 + Y.^2}) ./ \sqrt{X.^2 + Y.^2};$

$\text{mesh}(X,Y,Z);$ hold on;
 $\text{contour}(X,Y,Z);$ hold off;

Bölüm - 8 DEEP LEARNING

1 soru gelecek.

Artificial Intelligence

Yapay zeka, insan olarak akıllı varlıkların zeki davranışlarını taklit eden yazılımlardır. İnsanların yorum yapma, sonuç çıkarma, geçmiş tecrübelerden öğrenme gibi insana mahsus zeki davranışlar oluşturulmaya çalışılır.

AI, çevresinde meydana gelen olaylardan öğrenen bu olay ve eylemleri algılayıp karar vermesi intelligent agent dir.

Machine Learning

Bazı durumlarda karşılaştığımız problemi, çözümünü oluşturarak kadar algoritmasını, çıkartarak kadar iyi biliyoruz olabiliriz. Fakat elimizde birçok data var.

Training datayı kullanarak ilgilendiğimiz problemin bir matematiksel modelini üreten algoritmalaradır. "Prediction ve Decision"

Deep Learning ile farkı;

ML'de elimizdeki inputları kullanarak bir feature set çıkarıyor.
Bu featureları kullanarak bir model oluşturmuyoruz.

Deep Learning'de ise daha fazla hidden layerlerden oluşan ağların
Verilen inputa göre feature extraction ve classification işlemini de
yapar.

Deep Learning

Çok sayıda katman kullanarak yüksek seviyeli featurelar çıkarabilir.

Öğrenilen ağı kullanılır.

ML \rightarrow spam, hzm filter.

1943 - 2006 Prehistory of Deep Learning

1943 \rightarrow AND OR neuron

1958 \rightarrow neuron perception, binary sınıflandırma problemi.

1969 \rightarrow Perceptronlar XOR problemi çözemez.

1990 \rightarrow multilayer perceptronlar ile XOR problemi çözülebiliyordu.

Moravec's Paradox = ilk zklz gelenin aksine, karar verme karmaşık
hesaplar yapma işlemleri pc'ye yaptırmak kolaydır
ve az maliyetlidir. sensörlü dış dünyayı algılamak
problemleri çok ciddi beceri gerektirir.

Bu tür becerileri robotlara kazandırmak zordur.

Neden fazl oldu?

ImageNet yarışması ile popüler hale gelmeye başladı. (2009)
Alexnet var.

Neden deep learning popüler oldu?

- Verisetlerinin boyutunun artması
- New deep learning models
- Increasing computational power

MATLAB'de;

DNN modellerinden yaygın olarak kullanılan özellikli görüntü işleme modelleriyle kullanılan evrimsel sınırlı ağıdır. (Convolutional neural network.)

Görüntü verilerinde evrim işlemi uygulayarak özellikli hata oranı çıkarılır.
Bu özellikli hata oranı ortalamaya katmanını kullanarak tam bağlantılı (fully connected) katmanına yollar.

Evrimsel Katman

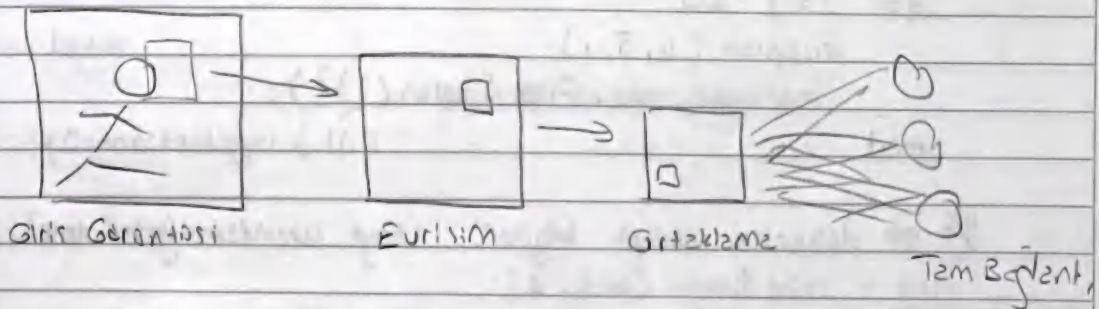
İ giriş görüntüsünün bir filtre aracılığıyla tırar, filtrenin nasıl geçeceğini parametre olarak verebiliriz.

Ortalama Katman (POOL)

Büyük miktardaki verimlerden özet çıkarma işlemini sağlar.

max-pooling → bulunmuş olduğu bölümden max değeri döndürür.

average-pooling → bulunmuş olduğu bölümden değerlerin ortalamasını bulur.



Tam Beğlenti

Yazay sınırlar zı modelidir. Sonuç çıkarmak için kullanılır.

Aktivasyon Fonksiyonları.

Elde edilen verilerin Doğrusal olmasını engellemek amacıyla aktivasyon fonksiyonları kullanılır. ReLU Sırtı ReLU ELU

Softmax → Sınıflandırma tabi uygulamalarda kullanılır.

ör

script açılır.

% % datasetin yüklenmesi

Butz'ı
Sarmaz

```
digitDatasetPath = fullfile(matlabroot, 'toolbox', 'nnet', 'nndemos',  
    'nndatsets', 'Digit Dataset');  
imds = imageDatastore(digitDatasetPath,  
    'IncludeSubfolders', true, 'LabelSource', 'foldernames');
```

% % datasetten örnek görüntülerin gösterilmesi

figure;

perm = randperm(10000, 20);

for i = 1:20

subplot(4, 5, i);

imshow(imds.Files{perm(i)});

end

% % datasetle ilgili bilgiler; (hangı resimden kaçar true)

img = readImage(imds, 1);

size(img)

labelCount = countEachLabel(imds)

% 90 Veri kümesini train ve test olarak ayırmak % 35 → train

numTrainFiles = 750;

[imdsTrain, imdsValidation] = splitEachLabel(imds, numTrainFiles, 'randomize');

% 90 Kütmanların kod ile oluşturulması → Deep Network Designer ile yapılabilir

layers = [
imageInputLayer([28 28 1]) → kenar sayısı RGB olsaydı → 3 olurdu
convolution2dLayer(3, 8, 'padding', 'same') → pencere boyutu → doldurama yap (kenarlar için)
→ filtre sayısı → gelen veri ile çıkan verinin boyutu eşit olsun.

batchNormalizationLayer → genelde Conv layer ile aktivasyon fonk
zsinde kullanılır.
Eğitimi hızlandırır.

reluLayer → aktivasyon fonk

maxPooling2dLayer(2, 'Stride', 2) → pencere boyutu → azaltma miktarı
→ Veriyi daha küçük boyuta indirir.

convolution2dLayer(3, 16, 'padding', 'same')

batchNormalizationLayer

reluLayer

fullyConnectedLayer(10) → 10 sınıfı temsil ediyor

softmaxLayer

classificationLayer;

% % Eğitim parametrelerinin yazılması

```
options = trainingOptions('sgdm', ...  
    'InitialLearnRate', 0.01, ...  
    'MaxEpochs', 4, ...  
    'Shuffle', 'every-epoch', ...  
    'ValidationData', imdsValidation, ...  
    'ValidationFrequency', 30, ...  
    'Verbose', false, ...  
    'Plots', 'training-progress');
```

→ Başlangıç öğrenme oranı (Başlangıçta öğrenir, hemen öğrenir.)
→ Tüm verinin elden geçirme 4 kere
→ veriyi karıştırılır.
→ test yapılacak veri belirtilir.
→ Kaç adımda bir gösterileceğimizi.
→ detaylı.
→ training aşamasını grafiklerle gösterir.

% % Networkun eğitilmesi

```
net = trainNetwork(imdsTrain, layers, options);
```

90 Bunu çalıştırınız

% Validation accuracy %98 çıktı.

% % Validation Doğruluk Değerinin Hesaplanması

```
YPred = classify(net, imdsValidation);  
YValidation = imdsValidation.Labels;
```

→ sınıflandırılacak
→ sınıflandırılacak veri

```
accuracy1 = sum(YPred == YValidation) / numel(YValidation)
```

% % Rastgele bir görüntünün eğitilen network ile tahmin edilmesi

```
randNum = randperm(2500, 1);  
randomImage = readImage(imdsValidation, randNum);  
classify(net, randomImage);  
imshow(randomImage);
```

→ bu tahmin eder çıktıdır.

Alexnet (Transfer learning)

Derin öğrenme modelleri genellikle büyük veri gerektirir. Ancak yeterli veri yoksa hazır modeller var. Bu modellerin eğitilmiş katmanlarını kullanabiliriz.

```
unzip('MerchData.zip');  
imds = imageDatastore('MerchData', ---  
    'IncludeSubfolders', true, ---  
    'LabelSource', 'foldernames');
```

```
[imdsTrain, imdsValidation] = splitEachLabel(imds, 0.7, 'randomized');
```

```
numTrainImages = numel(imdsTrain.Labels);
```

```
idx = randperm(numTrainImages, 16);
```

```
figure;
```

```
for i = 1:16
```

```
    subplot(4,4,i)
```

```
    I = readImage(imdsTrain, idx(i));
```

```
    imshow(I);
```

```
end
```

```
net = alexnet;
```

```
net.Layers.
```

```
inputSize = net.Layers(1).InputSize
```

→ Son 3 kısım atılır. (23'den sonrasını atılır.)

```
layersTransfer = net.Layers(1:end-3);
```

```
numClasses = numel(categories(imdsTrain.Labels));
```

```
layers = [
```

```
    layersTransfer
```

```
    fullyConnectedLayer(numClasses, 'WeightLearnRateFactor', 20,  
        'BiasLearnRateFactor', 20)
```

```
    softmaxLayer
```

```
    classificationLayer];
```


% image augementer, → Elimitdeki veriyi arttırma işlemidir.

pixelRange = [-30, 30];

imageAugementer = ImageDataAugementer (---
'Rand X Reflection', true, ---
'Rand X Translation', pixelRange, ---
'Rand Y Translation', pixelRange);

augImdsTrain = augmentedImageDatastore (inputSize(1:2), imdsTrain, ---
'Data Augmentation', imageAugementer);

augImdsValidation = augmentedImageDatastore (inputSize(1:2),
imdsValidation);

options = trainingOptions ('sgdm', ---
'MiniBatchSize', 10, ---
'MaxEpochs', 6, ---
'InitialLearnRate', 1e-4, ---
'ValidationData', augImdsValidation, ---
'ValidationFrequency', 3, ---
'ValidationPatience', Inf, ---
'Verbose', false, ---
'Plots', 'training-progress');

netTransfer = trainNetwork (augImdsTrain, layers, options);

[YPred, scores] = classify (netTransfer, augImdsValidation);


```
idx = randperm (numel (imds Validation. Files), 4);  
figure;  
for i = 1:4  
    subplot (2,2,i);  
    I = readImage (imds Validation, idx(i));  
    imshow (I);  
    lbl = YPred (idx(i));  
    title (string (lbl));  
end
```

```
Y Validation = imds Validation. Labels;  
accuracy = mean (YPred == Y Validation)
```